

Grado en Ingeniería Telemática
2016-2017

Trabajo Fin de Grado

“Predicción de resultados deportivos con técnicas de Machine Learning aplicado al fútbol”

Sergio Calderón Pérez-Lozao

Tutor

Jesus Cid Sueiro

Junio 2017



Índice

1 Extended Abstract	1
1.1 Data Cleaning	3
1.2 Model Selection.....	3
1.3 Prediction	4
1.4 Comparison	5
1.5 Conclusions	5
1.6 Future Work	6
2 Motivación.....	8
3 Objetivos	9
4 Estado del arte.....	11
4.1 FiveThirtyEight.....	12
4.2 Microsoft (Bing)	14
4.3 Google	16
5 Desarrollo	18
5.1 Obtención de los datos	18
5.2 Primeros pasos.....	18
5.2.1 Librerías	19
5.2.2 Dataset.....	20
5.2.3 Exploración de los datos.....	22
5.3 Machine Learning	29
5.4 Aprendizaje supervisado	31
5.5 Clasificación.....	34
5.5.1 Decision Tree.....	35
5.5.2 SVM.....	36
5.5.3 KNN.....	37
5.5.4 Logistic Regression	38
5.6 Regresión	39
5.6.1 Linear Regression	40
5.7 Aprendizaje no supervisado	40
5.7.1 Clustering para mejora de prestaciones	41
5.7.2 Modelo simple.....	42
5.7.3 Modelo desagregado	48
6 Resultados	56
6.1 Evaluación experimental	56

6.1.1 Parámetros de los algoritmos	56
6.1.2 Comparación	57
6.1.3 Selección de variables	59
6.2 Análisis de los resultados.....	61
6.2.1 Comportamiento	61
6.3 Redes Neuronales.....	69
7 Presupuesto y planificación.....	72
8 Entorno socio-económico.....	75
9 Marco regulatorio.....	76
9.1 Legislación aplicable.....	76
9.2 Estándares técnicos.....	76
9.3 Propiedad intelectual	76
10 Conclusiones.....	77
11 Bibliografía.....	79

Ilustraciones

Ilustración 1: Ejemplo de predicción y métricas del modelo de FiveThirtyEight	13
Ilustración 2: Evolución de la ventaja del equipo local en el fútbol inglés [3]	14
Ilustración 3: Predicción de Bing para la temporada 2016/17 de la liga española	15
Ilustración 4: Diagrama de bloques de la limpieza de los datos, generado con draw.io [18]	22
Ilustración 5: Histogramas de los ratings para cada una de las posiciones.....	23
Ilustración 6: Tabla que contiene el porcentaje de jugadores por encima o por debajo de un umbral determinado en función de la puntuación.....	24
Ilustración 7: Gráfico circular de los porcentajes de cada resultado	25
Ilustración 8: Visualización con Gephi de las diferentes ligas	26
Ilustración 9: Visualización de los partidos de la liga española dependiendo de los ratings de ambos equipos.....	27
Ilustración 10: Matriz de diagramas de dispersión de todos los pares de variables e histograma de cada variable	28
Ilustración 11: Visualización de los resultados a través de los ratings de los equipos y de las cuotas de las casas de apuestas.....	29
Ilustración 12: Cheat sheet de scikit-learn [21].....	30
Ilustración 13: Ejemplo de sobreajuste	33
Ilustración 14: Compensación entre sesgo y varianza [22]	33
Ilustración 15: Precisión frente a exactitud	34
Ilustración 16: Ejemplo de árbol de decisión aplicado a clasificación de viviendas en Nueva York o San Francisco [24]	35
Ilustración 17: Ejemplo de SVM [26].....	37
Ilustración 18: Ejemplo de KNN [28].....	38
Ilustración 19: Ejemplo de regresión logística [30]	39
Ilustración 20: Ejemplo de regresión lineal aplicado al famoso caso de moneyball [32]	40
Ilustración 21: Ejemplo de aplicación del estudio propuesto	42
Ilustración 22: Visualización de los pasos que sigue k-means para encontrar clusters en los datos [35]	44
Ilustración 23: Clusters de k-means (colores), visualizado por calidad de los equipos.....	45
Ilustración 24: Clusters de k-means (colores), visualizado por cuotas de las casas de apuestas ...	46
Ilustración 25: Visualización del algoritmo DBSCAN encontrando los grupos por densidades	47
Ilustración 26: Visualización de los grupos que forma DBSCAN por equipos	48
Ilustración 27: Matriz de diagrama de dispersión e histogramas en la diagonal para las 40 variables	52
Ilustración 28: Método del codo para elegir el número de clusters en k-means.....	53
Ilustración 29: Grupos de variables para k-means	54
Ilustración 30: Grupos de variables para DBSCAN	55
Ilustración 31: Ejemplo de código que inicializa los objetos de los algoritmos utilizados de aprendizaje supervisado	57
Ilustración 32: Ejemplo del código para la métrica ideada aplicada a la regresión lineal.....	58
Ilustración 33: Tabla comparativa entre los diferentes algoritmos de aprendizaje supervisado, generado con draw.io [18].....	59
Ilustración 34: Matriz de dispersión e histogramas de las variables después de realizar la selección	60

Ilustración 35: Pequeña muestra de partidos en los que el equipo local era superior, resultó ganador y nuestro modelo acertó.....	62
Ilustración 36: Pequeña muestra de partidos en los que el equipo local era superior, no resultó ganador y nuestro modelo decidió que ganaría	63
Ilustración 37: Pequeña muestra de partidos en los que el equipo visitante era superior, resultó perdedor y nuestro modelo acertó.....	63
Ilustración 38: Pequeña muestra de partidos en los que el equipo visitante era superior, resultó ganador o empataron y nuestro modelo no acertó.....	64
Ilustración 39: Probabilidad de que gane el equipo local según las casas de apuestas y el modelo enfrentadas	64
Ilustración 40: Probabilidad de que haya empate según las casas de apuestas y el modelo enfrentadas	65
Ilustración 41: Probabilidad de que gane el equipo visitante según las casas de apuestas y el modelo enfrentadas	65
Ilustración 42: Aciertos y errores del modelo y de las casas de apuestas en el subconjunto de validación para el caso en el que ganó el equipo local	66
Ilustración 43: Aciertos y errores del modelo y de las casas de apuestas en el subconjunto de validación para el caso en el que hubo empate	67
Ilustración 44: Aciertos y errores del modelo y de las casas de apuestas en el subconjunto de validación para el caso en el que ganó el equipo local	67
Ilustración 45: Cheat sheet de los tipos de redes neuronales que existen [50]	70
Ilustración 46: Representación de la red neuronal implementada, generado con draw.io [18]	71
Ilustración 47: Últimas iteraciones del entrenamiento de la red neuronal.....	71
Ilustración 48: Presupuesto del proyecto	73
Ilustración 49: Digrama de Gantt del proyecto	74

1 Extended Abstract

Soccer is one of the most critical sports with science and that is the main reason we develop this Bachelor Thesis. We want to analyze football data and try to infer its behavior and get its insights. We could apply this work to predict results in any sport, but we will focus on soccer for the entire project.

We will develop a Python [1] based system that gathers the data from an open database and analyzes it with different open source libraries. This system will be capable of predicting soccer results with a reasonable accuracy and we will compare it with the sports betting.

Our technique will consist on using machine learning algorithms that can be useful for this kind of study and are one of the most popular approaches at this moment. We will implement many of the most used algorithms not only from supervised learning, which is the natural way to face this problem, but also from unsupervised learning.

After comparing all of these algorithms we will analyze in depth the results and get conclusions for this research thesis.

1.1 Objectives

The main target in this bachelor thesis is analyzing a real-world problem like sports games with a data based methodology. Our objective is not to develop a final application, but to carry out a research work on different machine learning techniques that exist and be able to apply them to our problem.

Combining different skills that have been obtained throughout the whole degree of telematic engineering is another of the objectives, since it seeks to apply part of that knowledge to a project of these characteristics.

We want to do everything with open source technologies including the dataset, programming language and machine learning libraries, making it accessible to everyone and easily reproducible.

Because this is a bachelor thesis, another one of the principal goals is learning about all of these technologies and be capable of obtaining useful insights.

1.2 Stages

In order to build a good strategy to develop this project we have to divide our problem in four points:

- Data extraction: The database that we have chosen is in sqlite format, so we will have to use sqlite language to manage that information and be able to process it.
- Data cleaning: Once we have all the data in csv files -that is the format that we have taken as a standard for our study- we have to transform the raw data into valid data in order to start with machine learning tasks. To make this step we will use pandas [2], a Python library that facilitates this job.
- Modeling: One of the most important steps in our bachelor thesis, we have to choose between the different models and applying them to the data. For this phase, we will use scikit-learn [3] library, also for python, which is very powerful for implementing machine learning algorithms. For the numerical calculations, we will use numpy [4], one of the most famous libraries in Python for scientists.
- Visualization: So as to analyze the results and make comparisons between them, visualizations are very useful and for this purpose we will implement them with matplotlib [5] and seaborn [6], two visualization libraries for Python, the first is more general and the second is focused on statistical visualization.

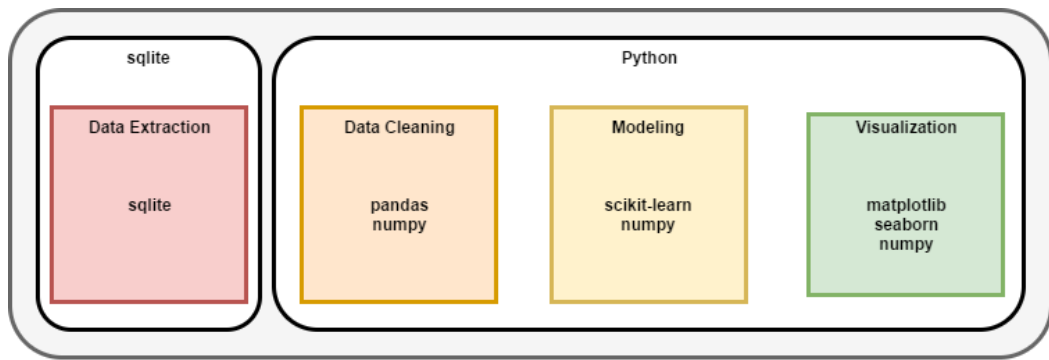


Ilustración 1: Tecnologías utilizadas en la elaboración del proyecto, generado con draw.io [7]

1.3 Data Cleaning

First of all, when we have all of the available data, it's necessary to clean it because of null registers or data that we won't use at all. For this task, we will use pandas [2] library for Python programming language, which makes it much easier for us.

Although this is the least visible part of the project, it's one of the costliest in terms of time, since it is necessary to devise data cleaning strategies and deal with a lot of information.

After cleaning the null data and making decisions for feature selection we obtain different data from matches, teams and players that we will use for our purpose. All of this data has to be analyzed before continuing the work, because it is very important to have smart data to get good results.

1.4 Model Selection

We have to choose a model to train it with the data from the previous step. We will apply supervised and unsupervised learning algorithms, comparing all of them to get the best one for this data input and the outcome we are searching.

With Python, we manage the data in pandas dataframes [8] and using numpy [4] for numerical calculations. In order to make the model selection, we will use scikit-learn [3], which makes easier applying machine learning algorithms.

For unsupervised learning, we have been inspired by a research paper called "The Utility of Clustering in Prediction Tasks" [9] to try to improve the accuracy with clustering techniques.

After comparing all of the models that we have implemented with different metrics, the logistic regression is the one who fits best to our data. We can't improve our accuracy for the algorithms that we have chosen, so we will make predictions with logistic regression model and we will compare it with bets' odds.

1.5 Prediction

For every match given we can make a prediction about which team will win with an associate probability. The model that fits better the data will return probabilities for each class of our multiclass problem: local team wins, draw or away team gets the victory.

These probabilities will be bias because of the home ground advantage in football that the model learns from the historical data, but that doesn't meant that our prediction is wrong. Furthermore, our model learns from players' ratings data and bets odds, making a strong prediction and based on data.

When we will make a prediction, we will compare its accuracy with two thresholds, the minimum threshold is the one that predicts that the home team will always win and has an accuracy of 46%. The second threshold is the sports betting based and has an accuracy of 53%.

If a prediction has an accuracy below the minimum threshold we discard it, but if it has an accuracy above the minimum threshold we consider that the algorithm has learned to predict by obtaining a better accuracy that is above the minimum

one. If our model surpasses the maximum threshold, then we consider that we have obtained a very good result.

One very important thing to keep in mind is that football is one of the most unpredictable sports in the world, so it is very difficult to make a safe prediction that, for example, give us an 80% (or more) chance of winning.

1.6 Comparison

Predictions will be made on the dataset, splitting it in training and test examples, so we know the final result of every match that we will predict.

We will compare betting odds with our prediction model, checking all cases when both of them predict correctly, both are wrong or one is right and the other one is wrong.

In this section, we will realize that our model works pretty well and it is an evidence that this very simple model can be very useful for this kind of project.

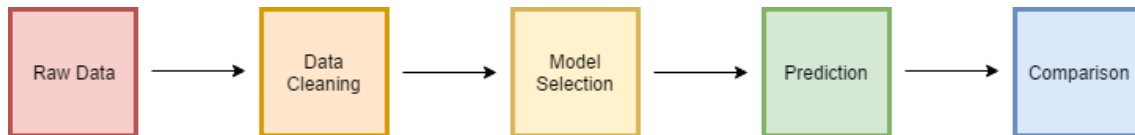


Ilustración 2: Flujo de información, generado con draw.io [7]

1.7 Conclusions

We can see that using data from soccer, or any sport, can help us understand it better. Inferring human behavior is not easy, but we can get close if we study and analyze data based decisions.

More and more data is being gathered over the years and this is not only in sports but also in many other disciplines. Advanced statistics methods like machine

learning are increasing in popularity and it can be very useful to research these types of techniques.

For this bachelor thesis, we study the importance of having a good dataset, cleaning process and, finally, choosing a good model to fit the data. It is obviously that if we get more smart data we will be able to make better predictions, so it's very important collecting the right data and always being looking for new possibilities like geopositional, climate or social data, to put several examples.

Also, we are conscious that this is not a perfect job, maybe the model is quite simple and we could apply more elaborated model or techniques that exists nowadays, but we have to know that we spent a lot of time doing training courses or learning about new algorithms or data to attack the problem.

We should never stop investigating new sources of data and models with which to train them and if it's possible make them open source technologies, because a lot of people together can do wonderful things.

"It's unbelievable how much you don't know about the game you've been playing all your life." Said Mickey Mantle, one of the best baseball players of all time.

1.8 Future Work

Taking into account several details that we have mentioned before, it will be very convenient to look for another data source with which we can improve our accuracy. If we consider this new way, we will have to compare again all of the algorithms and think about new models or machine learning techniques that may fit better with the new data.

In a real time application that deals with this kind of problems, predictions should always be made with past data and it should predict results in a future historical period. We will not do this kind of approach in this bachelor thesis, but if we

would want to use this project to face a real-world application, we have to change the way that we will split the data.

Besides looking for new data sources and give it a real-time approach, we can research more complex methods of machine learning to learn from the data and enhance our performance. Neural networks are one of the most attractive manners to infer behavior from data when we have many features and data, so it will be helpful applying it to our data.

2 Motivación

El fútbol es un deporte donde siempre se ha dicho que dos más dos nunca suman cuatro, es decir, que en este deporte la ciencia no tiene cabida. En otros deportes se ha utilizado la estadística desde hace años para intentar predecir los resultados finales.

La primera muestra de la aplicación de esta disciplina en el deporte se remonta al año 2002, cuando los Athletics de Oakland, después de una etapa de crisis el año anterior, necesitaban formar un equipo de béisbol competitivo con un presupuesto mucho menor que el resto de sus rivales. Decidieron basarse en la estadística avanzada para crear nuevo equipo a partir de la incorporación de un grupo lleno de figuras infravaloradas por el mercado.

A este método se le llamó Moneyball [10] y con ello consiguieron 20 victorias consecutivas, lo que nunca nadie antes había logrado. Este hito histórico hizo que los demás equipos empezaran a tener en cuenta estas técnicas a la hora de contratar deportistas y analizar sus jugadas.

Hoy en día todos los equipos de béisbol tienen un departamento de ciencia de datos, donde calculan métricas aplicadas que les pueden ayudar en la toma de decisiones. Esta técnica cruzó las barreras del béisbol y se empezó a utilizar para estudiar los demás deportes. El fútbol ha sido una de las disciplinas más complicadas de analizar, por lo que aún queda mucho más por descubrir.

Es un deporte muy difícil de predecir y resulta un desafío importante estudiar algunas de las formas existentes para modelar su comportamiento. Uno de los equipos de fútbol que utilizó estas técnicas avanzadas de estadística y Big Data fue el Leicester, que consiguió ganar la Premier League justo un año después de subir desde la segunda división inglesa.

3 Objetivos

El objetivo principal de este proyecto es explorar el problema de predicción de resultados de fútbol mediante técnicas de aprendizaje automático (Machine Learning), al analizar el comportamiento de varios algoritmos en el estado del arte dándoles como aplicación la predicción de partidos de fútbol. Para ello será necesario encontrar un conjunto de datos libres que resulten útiles para dicha tarea.

Con este estudio no se pretende mejorar el rendimiento de las actuales soluciones que existen, ya que no se dispone ni de los recursos ni del tiempo necesarios para ello. Se trata de hacer una comparación entre los diferentes tipos de aprendizaje y algoritmos para cada uno de ellos, para poder conocer más resultados acerca de los análisis estadísticos que se podrían utilizar para demostrar que en el fútbol hay factores que pueden tener más peso que otros a la hora de lograr que equipo se alce con la victoria.

El proyecto será desarrollado íntegramente en Python y, por tanto, otro de los objetivos consistirá en conocer la potencia de este lenguaje para una labor como la que se presenta y familiarizarse con las diferentes librerías que existen para ello.

Además, todas las tecnologías utilizadas en el estudio son libres y están a disposición de cualquier persona que quiera acceder a ellas gratuitamente, tanto las fuentes de datos como las librerías utilizadas en la parte del desarrollo. Este proyecto tiene un fin exploratorio, con recursos libres y, por tanto, fácilmente reproducible.

La estructura de esta memoria a partir de este punto comenzará revisando el estado del arte, dedicando un capítulo completo a los usos de los diferentes algoritmos que se utilizarán en el estudio y a casos concretos de predicción aplicados al fútbol. A continuación, se presentará el desarrollo, dividido en dos partes, aprendizaje supervisado y no supervisado, donde se repasan todas las

comparaciones entre los algoritmos estudiados y se profundiza en el problema planteado y en sus posibles soluciones. Por último, se analizarán los resultados y se presentarán las conclusiones del estudio.

4 Estado del arte

Utilizar técnicas de Machine Learning para el mundo del deporte está en auge. Desde hace años se ha invertido mucho tiempo en investigación para intentar encontrar la forma más precisa de predecir este tipo de eventos, pero es ahora, gracias a la enorme cantidad de datos existente, cuando se hace algo más accesible.

En lo que se refiere al fútbol, que es el deporte que se va a abordar en este proyecto, ha empezado hace relativamente poco, pero gracias a la gran cantidad de dinero que maneja y el gran interés del público en general, se está invirtiendo muchos recursos en mejorar los modelos para la predicción de resultados.

Existen multitud de fuentes de datos que tener en cuenta y poder utilizar para poder fortalecer nuestras predicciones sobre un partido concreto. No solo datos de enfrentamientos anteriores, también, por ejemplo, el clima, las conversaciones en redes sociales, etcétera.

A la pregunta sobre como modelar matemáticamente un partido de fútbol existen muchos modelos actualmente que funcionan relativamente bien y son completamente diferentes.

Se sabe que los goles de un partido de fútbol pueden modelarse con un proceso de Poisson y muchos de los estudios actuales se basan en ello, generalmente con una métrica para los goles esperados por un partido, según la fuerza de ambos equipos.

El principal problema, y de ahí la gran dificultad que existe para predecir correctamente un partido de fútbol, es que el marcador es muy ajustado, es decir, un equipo fuerte puede tener 2 goles esperados y uno débil 1, haciendo que exista una probabilidad razonable de que gane cualquiera de los dos.

En este apartado nos centraremos en el modelo de FiveThirtyEight, ya que es el modelo más completo que existe en la actualidad cuya metodología esté publicada. También haremos énfasis en los casos de Google y Bing, con una metodología más cerrada, pero de los que se conocen ciertos detalles.

Los ejemplos son de tipo aplicación, contrariamente a nuestro estudio, el cual tiene una finalidad mucho más exploratoria y de investigación, aunque el esqueleto de estas aplicaciones es un análisis estadístico como el que se presenta en esta memoria, por lo que se puede analizar el estado del arte de esta forma.

4.1 FiveThirtyEight

A día de hoy el modelo que más llama la atención es el de FiveThirtyEight [11], el medio que utiliza periodismo de datos más popular de la red. Resumiendo, utiliza una puntuación para cada equipo llamado SPI, ideada por el propio Nate Silver (creador de FiveThirtyEight) en 2009.

El SPI es, básicamente, el resultado que obtendría un equipo jugando contra un equipo medio. Por tanto, son dos números, uno ofensivo y otro defensivo donde el primero será más poderoso cuanto más alto y el segundo todo lo contrario, ya que eso significará que el conjunto en cuestión es capaz de meterle muchos goles y encajar pocos contra un equipo de calidad regular.

Con estos ratings ya se pasa a hacer la predicción. Después de cada partido los ratings se reajustan teniendo en cuenta el rendimiento y la fuerza del rival. Si un equipo gana, pero lo hace por menos de lo esperado el rating disminuye, al contrario que Elo.

Podemos ver como partido de ejemplo el Barcelona – Real Madrid de esta temporada, disputado en el mes de diciembre. En verde está coloreado el resultado que finalmente se dio y tenemos las probabilidades para cada posible

signo de la quiniela. Debajo del resultado nos muestran las métricas con las que reajustan los ratings después del partido.



3/12	 Barcelona 1	64%	18%
	 Real Madrid 1	18%	
		BAR	MAD
Adjusted goals		1 . 1	1 . 1
Shot-based xG		1 . 9	1 . 5
Non-shot xG		1 . 7	2 . 0

Ilustración 3: Ejemplo de predicción y métricas del modelo de FiveThirtyEight

No nos dan datos exactos sobre sus precisiones, aunque podemos ver en su página web las predicciones en cuanto a porcentaje de victoria, derrota o empate para cada partido. Por lo general cuando el porcentaje es mayor del 80% suele acertar en su pronóstico, aunque no goza de una precisión exquisita.

Explorando un poco las predicciones de FiveThirtyEight observamos algo que vamos a ratificar después en nuestro estudio: El empate es el resultado más impredecible del fútbol sin lugar a dudas.

El equipo de Nate Silver se basa en un modelo de Poisson, donde los parámetros son el rating SPI comentado anteriormente, la ventaja de jugar en local y el número de días de descanso de cada equipo.

FiveThirtyEight destaca que la ventaja de jugar como equipo local no es tan determinante como hace años, aunque lo sigue siendo. Por ello, no es un valor estático, sino que es variable y decreciente en el tiempo, como podemos ver en la siguiente gráfica:

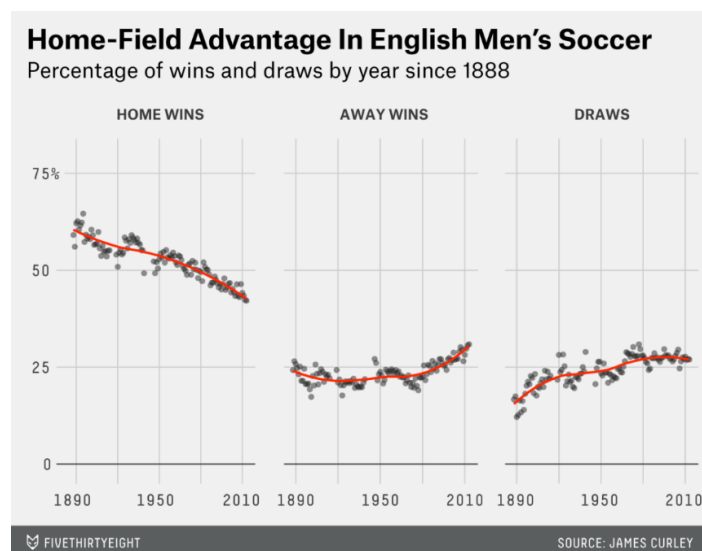


Ilustración 4: Evolución de la ventaja del equipo local en el fútbol inglés [12]

Con este modelo de Poisson pueden sacar una estimación de los goles de cada equipo y así computar la probabilidad de victoria de un equipo sobre otro o incluso el resultado concreto más probable.

Lo realizan aplicando el método de Monte Carlo, repitiendo el experimento 10000 veces, variando los ratings dependiendo de los resultados de sus pronósticos.

Su metodología completa está publicada en su web [13] y para obtener los datos utilizan Opta [14], un proveedor de datos deportivos muy completo que no se encuentra disponible de forma libre.

4.2 Microsoft (Bing)

El popular buscador de Microsoft, Bing [15], también apuesta por el análisis estadístico para predecir partidos de fútbol. La gran diferencia de este modelo es que utiliza la percepción de los usuarios a través de las búsquedas en su plataforma y datos de redes sociales.

Bing utiliza el Big Data y algoritmos de Machine Learning que son entrenados para realizar las predicciones finales en forma de probabilidades. Las precisiones totales, al igual que en el modelo de FiveThirtyEight, no están publicadas, aunque pueden presumir de conseguir un 15 de 16 en las fases eliminatorias del pasado

mundial, superando en un partido al modelo que ideó Google, del que hablaremos a continuación.

Cabe destacar que no se puede usar en crudo los datos de la popularidad de un equipo en una conversación determinada o a lo largo del tiempo, ya que existen bastantes sesgos emocionales. Hay equipos más queridos que otros o más populares, por lo que Bing tiene en cuenta el análisis del sentimiento de esos datos y así ponderar, por ejemplo, según objetividad.

También les diferencia el hecho de que utilizan la meteorología para cada partido como parte de los datos de entrenamiento que se le pasan a los modelos.

La metodología no está expuesta como la de FiveThirtyEight, pero sabemos sus pronósticos para esta temporada de la liga Bing:

Posición	Equipo	Puntos
1	Real Madrid Club de Fútbol	96
2	Fútbol Club Barcelona	95
3	Club Atlético de Madrid	88
4	Sevilla Fútbol Club	70
5	Athletic Club	68
6	Real Club Celta de Vigo	64
7	Villarreal Club de Fútbol	64
8	Unión Deportiva Las Palmas	63
9	Málaga Club de Fútbol	62
10	Valencia Club de Fútbol	60
11	Real Betis	58
12	Sociedad Deportiva Eibar	56
13	Real Sociedad de Fútbol	54
14	Club Deportivo Leganés	40
15	Deportivo Alavés	38
16	Real Club Deportivo Español	33
17	Real Sporting de Gijón	31
18	Real Club Deportivo de La Coruña	29
19	Granada Club de Fútbol	21
20	Club Atlético Osasuna	17

Ilustración 5: Predicción de Bing para la temporada 2016/17 de la liga española

4.3 Google

El buscador más popular también tiene su propio modelo para predecir el resultado de partidos de fútbol. Los datos los obtiene, al igual que FiveThirtyEight, de Opta, y no agrega características sociales como si hace Microsoft.

Google utiliza como algoritmo de Machine Learning la regresión logística, un modelo muy simple pero que funciona relativamente bien dada la dificultad del problema. Principalmente fue creado para el mundial de 2014 pero se puede llevar a un problema diferente.

Debido a que está pensado para las fases finales del mundial, donde se ha jugado previamente solo la fase de grupos, ya no hay empates. Esta regresión logística se aplica para los tres últimos partidos que se han jugado, es decir, utilizan un histórico de tres partidos, para sacar las probabilidades que tiene cada equipo de pasar a la siguiente ronda.

Con esos tres partidos se crea una matriz donde se tiene en cuenta que equipo es local y la diferencia de goles que ha habido entre ellos. De ahí se pueden calcular los pesos de cada equipo, es decir, lo fuerte que es cada equipo, con la regresión logística y poder predecir los partidos.

Públicamente lo utilizaron para el pasado mundial, donde consiguieron los ya mencionados 14 de 16 y han liberado el código en Github [16] para que cualquiera lo pueda utilizar y hacer sus propias predicciones, e incluso intentar mejorar el código.

Estos modelos y otros menos populares que existen sobre predicción en partidos de fútbol nos animan a seguir estudiando las posibilidades que existen para ser más precisos en nuestras apuestas y poder conocer el ganador antes de que se juegue el partido, sabiendo que nunca lo podremos conseguir al 100%.

Hay que tener en cuenta que una sola acción puede cambiar por completo un partido de fútbol, como un gol en los primeros minutos o la lesión del mejor de

los jugadores de un equipo. Por ello se está investigando en el uso de datos a tiempo real para la predicción de resultados, aunque requiere de cálculos mucho más costosos y técnicas de Big Data.

Lo más parecido que existe por el momento en los modelos de los que hemos hablado para el estado del arte son las simulaciones de Monte Carlo, donde se introducen muchas variaciones en los diferentes experimentos y se tienen en cuenta para realizar la predicción final.

Las técnicas de Machine Learning que se utilizan para predecir eventos son muy potentes, pero no son magia, saber interpretar los resultados y las probabilidades es clave a la hora de valorar los modelos.

5 Desarrollo

5.1 Obtención de los datos

Para la realización de este trabajo es necesario un histórico de partidos lo suficientemente grande como para poder hacer el análisis. Además de la cantidad de los datos necesitamos también que éstos sean de calidad, ya que, por muy bueno que sea el modelo con el que entrenemos los datos, si no son de calidad no nos será posible sacar buenas conclusiones de los mismos.

En Internet existen muchas fuentes de datos de este tipo, muchas de ellas son libres, en las cuales nos centraremos, y otras que no lo son. Dentro de las bases de datos de partidos de fútbol libres parece que la mejor opción será la 'European Soccer Database', ubicada en la red social de ciencia de datos Kaggle [17].

La razón por la cual esta base de datos parece la más idónea para realizar el trabajo es porque, además de incluir más de 25.000 partidos de las grandes ligas, atributos de jugadores, partidos y equipos, Kaggle posee una de las mayores comunidades de data scientists del mundo. Esto facilita mucho las cosas ya que cualquier tipo de duda con los datos puede ser resuelta en minutos, además de pequeños códigos en Python o R que nos permiten ver análisis de los datos que han realizado otras personas.

No obstante, hay que tener en cuenta las limitaciones que tiene este conjunto de datos.

5.2 Primeros pasos

La base de datos está en formato sqlite, por lo que habrá que utilizar queries de SQL para la obtención de los datos. También procedemos a escoger el lenguaje de programación en el que desarrollar nuestro estudio, principalmente dudamos entre R y Python, ya que son los dos lenguajes más utilizados en el mundo de la ciencia de datos.

Python es un lenguaje de propósito general, en cambio R es un lenguaje pensado para el análisis estadístico, por lo que es mucho más específico. El ámbito de R es mucho más cerrado, se suele utilizar en el campo de la investigación, aunque se está expandiendo rápidamente al mundo laboral. Python, en cambio, valora la productividad y la fácil lectura de su código, siendo más recomendable que R para desarrolladores.

5.2.1 Librerías

Finalmente, Python será el lenguaje elegido para realizar este trabajo, teniendo en cuenta además las excelentes librerías que posee en el campo del análisis estadístico y el Machine Learning. Las librerías elegidas para utilizar en este proyecto son las siguientes:

Para el análisis de los datos y su manipulación, utilizaremos una de las librerías más populares en Python para ello, pandas [2]. Con pandas tendremos estructuras de datos muy sencillas de manejar, lo que nos facilitará la tarea de transformaciones dentro de los datos y la limitación que podría suponer el uso de sqlite.

En lo que se refiere a los modelos de Machine Learning que vamos a utilizar nos apoyaremos de scikit-learn [3], también una de las librerías más utilizadas por la comunidad de Python para estos fines. Scikit-learn representa cada algoritmo en un objeto con métodos muy parecidos, teniendo como funciones principales el entrenamiento (fit) y la predicción (predict), lo que facilita mucho su uso.

Usaremos numpy [4] para todo tipo de cálculo numérico que necesitemos computar. Con esta librería nos será mucho más fácil realizar cualquier tipo de cálculo, además de poder utilizar los numpy.array [18] en sustitución de las listas nativas de Python, ya que éstas no soportan el cálculo vectorial.

Las conclusiones visuales de nuestro estudio las haremos con seaborn [6] o matplotlib [5], siendo la primera para visualizaciones con mayor carácter

estadístico, ya que contiene muchísimos métodos que sirven para visualizar distribuciones y modelos estadísticos, y la segunda para gráficos en general, aunque seaborn está implementada por encima de matplotlib, aportando una mayor abstracción a la hora de hacer los gráficos.

5.2.2 Dataset

Para entender un poco mejor el contexto del proyecto hay que conocer la estructura de la base de datos elegida para su realización. Este dataset libre cuenta con más de 25000 partidos y 10000 jugadores, desde 2008 hasta 2016.

La base de datos se estructura en 7 tablas:

- **Country:** Contiene los 11 países de los que provienen los datos, Bélgica, Inglaterra, Francia, Alemania, Italia, Holanda, Polonia, Portugal, Escocia, España y Suiza.
- **League:** Mapea los 11 países a sus respectivas ligas
- **Match:** Esta tabla contiene todos los partidos con la información de eventos como por ejemplo goles, faltas o porcentaje de posesión. Además de los 22 jugadores que jugaron el partido de inicio, la liga a la que pertenece y los goles de cada equipo.
- **Player:** Incluye los nombres de los jugadores, su peso y altura, vinculándolos con un id único para acceder a sus atributos.
- **Player_Attributes:** 39 columnas con diferentes características de los jugadores como definición de cara a puerta, pie favorito o reflejos de portero, además de la fecha de actualización del atributo y los identificadores para cruzar con la tabla de jugadores.
- **Team:** Identificadores únicos y nombres de cada equipo.
- **Team_Attributes:** Similar a la tabla de Player_Attributes pero con atributos como la capacidad de creación de juego o de crear una ocasión de peligro.

Esta base de datos es una agregación de 3 fuentes de datos de fútbol, la primera sobre eventos, alineaciones y marcadores [19], la segunda para las cuotas de las casas de apuestas [20] y por último los atributos de los jugadores y equipos [21].

Al enfrentarnos por primera vez a la base de datos vemos que tendremos que hacer una buena tarea de limpieza de nuestro dataset. Existen muchos tipos de atributos o eventos, que fueron introducidos en la base de datos mucho después de la fecha del primer partido, es decir, no se tienen datos de ese tipo de los primeros partidos. También faltan algunos datos sueltos o incluso existen muchos datos cuya fecha supera a la mayoría de partidos y por tanto tampoco podría utilizarse por coherencia.

Cuando nos encontramos con estos problemas tendremos que decidir qué hacer con nuestros datos. Las diferentes opciones que se barajan son, simplemente eliminar las columnas o filas que contengan mayor número de datos vacíos, asignarles un valor que sea la media de esas características a lo largo de todo el histórico, y, por último, predecir esos valores con algún tipo de modelo estadístico.

Si nuestro problema es que una característica contiene muchos nulos, lo mejor es eliminar dicha columna, ya que no nos va a aportar información útil para realizar nuestra predicción. Es el menor de los problemas que nos vamos a encontrar, ya que no afecta al número histórico de partidos, por tanto, ante este problema eliminamos dichas características.

Cuando el problema es que un partido tiene muchas características que no tienen ningún valor asociado es un caso un poco peor, porque eso hace que el partido en cuestión no se pueda tener en cuenta para el modelo, ya que un partido sin características no nos va a aportar información. Por tanto, cada fila que eliminemos hace que nuestro modelo se vaya a nutrir con un partido menos, pero la mejor solución sigue siendo eliminar las filas.

Por último, puede haber datos sueltos perdidos, los cuales sí podría ser beneficioso ponerles un valor con la media de esa característica o una predicción de ese atributo, pero después de estudiar la base de datos vemos que son casos muy residuales y por tanto también optaremos por eliminar dichas filas, perdemos partidos para el histórico, pero ganamos tiempo.

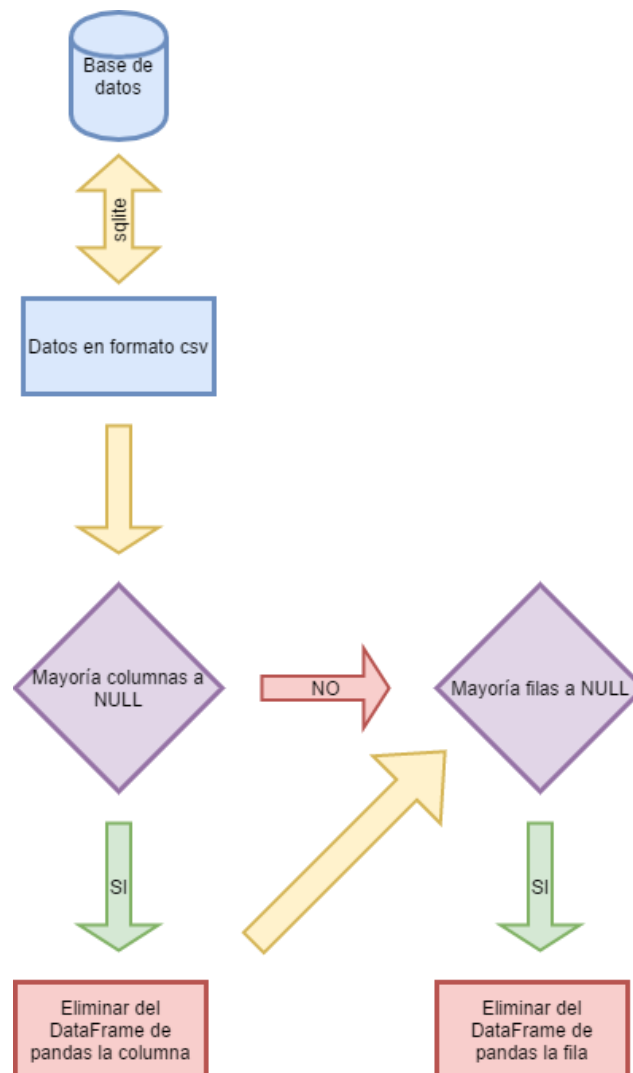


Ilustración 6: Diagrama de bloques de la limpieza de los datos, generado con draw.io [7]

Los datos serán obtenidos con consultas sql, convertidas a archivos csv para después tratarlos con pandas y poder realizar la limpieza en el propio código Python.

5.2.3 Exploración de los datos

Una vez hemos realizado la limpieza de los datos podemos visualizar las diferentes variables que vamos a utilizar para obtener una predicción. Para estas

primeras visualizaciones utilizaremos matplotlib, con el estilo de FiveThirtyEight, ya que es bastante limpio y atractivo.

En la base de datos vemos que tenemos puntuaciones asociadas a cada jugador en diferentes campos, como por ejemplo definición de cara a portería o capacidad defensiva. Existe un rating global que engloba todos los demás, siendo una buena métrica para evaluar la calidad de los jugadores.

El rating global es una puntuación de 1 a 100, cuanto más alto mayor calidad. Hacemos el análisis por posiciones de todos los jugadores de la base de datos, teniendo en cuenta siempre una formación 4-3-3 para tener en cuenta a un jugador como defensa, centrocampista o delantero.

Visualizamos los histogramas de los ratings globales para las diferentes posiciones, viendo cómo se distribuyen los valores para cada tipo de jugador.

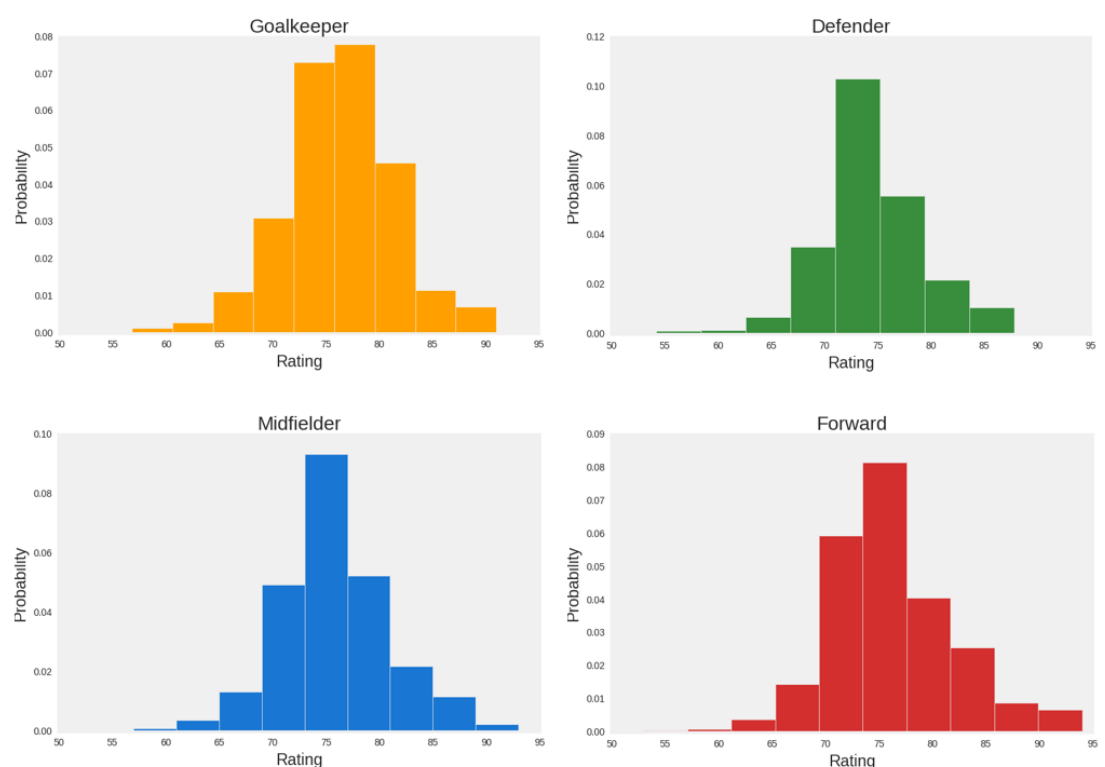


Ilustración 7: Histogramas de los ratings para cada una de las posiciones

Siguiendo los diferentes histogramas podemos observar como la mayoría de los jugadores se posicionan en los 70-80 de puntuación. También nos muestra como

las posiciones que contienen mayor frecuencia de ratings elevados son la de portero y los delanteros.

En los mediocentros hay más igualdad entre el número de jugadores que están por encima y por debajo de lo común, además de ser el tipo de jugador que más se suele situar en el umbral del 70% - 80%. Los defensas, por su parte, contienen más valores por debajo de la media que por encima.

	0% - 70%	70% - 80%	80% - 100%
Portero	8.9	67.4	23.7
Defensa	16.5	69.8	13.7
Centrocampista	11.8	72.7	15.5
Delantero	10.3	68.1	21.6

Ilustración 8: Tabla que contiene el porcentaje de jugadores por encima o por debajo de un umbral determinado en función de la puntuación

Una vez hecha una vista general de los jugadores vamos a proceder a analizar los partidos, empezando por un gráfico circular para los porcentajes de los diferentes resultados, viendo una clara tendencia del equipo local para ganar.

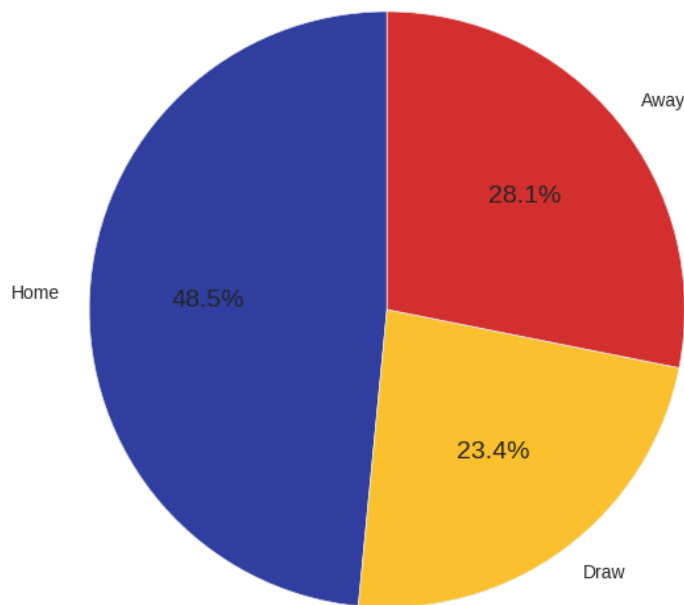


Ilustración 9: Gráfico circular de los porcentajes de cada resultado

Una tendencia esperada, que combinada con la fuerza de los equipos puede hacer que nuestros modelos aprendan fácilmente a predecir resultados de fútbol. Antes de pasar a una visualización de los partidos dependiendo de la calidad de los jugadores, vamos a utilizar gephi [22], un paquete de software para el análisis y visualización de redes a través de grafos.

Le pasaremos a gephi todos los partidos que tenemos, siendo cada nodo un equipo de fútbol y las aristas si ha habido un partido entre ambos equipos. Además, podemos visualizar más dimensiones con el tamaño de los nodos y su color, así como el color de las aristas.

En la siguiente imagen veremos la relación entre todos los equipos del dataset y la conexión entre ellos. El tamaño del nodo es proporcional a los puntos históricos relativos (durante todo el tiempo que tenemos disponible en los datos) y su color también es proporcional a los puntos. El algoritmo de layout utilizado para esta visualización es Fruchterman-Reingold [23] y sus parámetros son 10000 para el área, 10 de gravedad y 1 de velocidad.

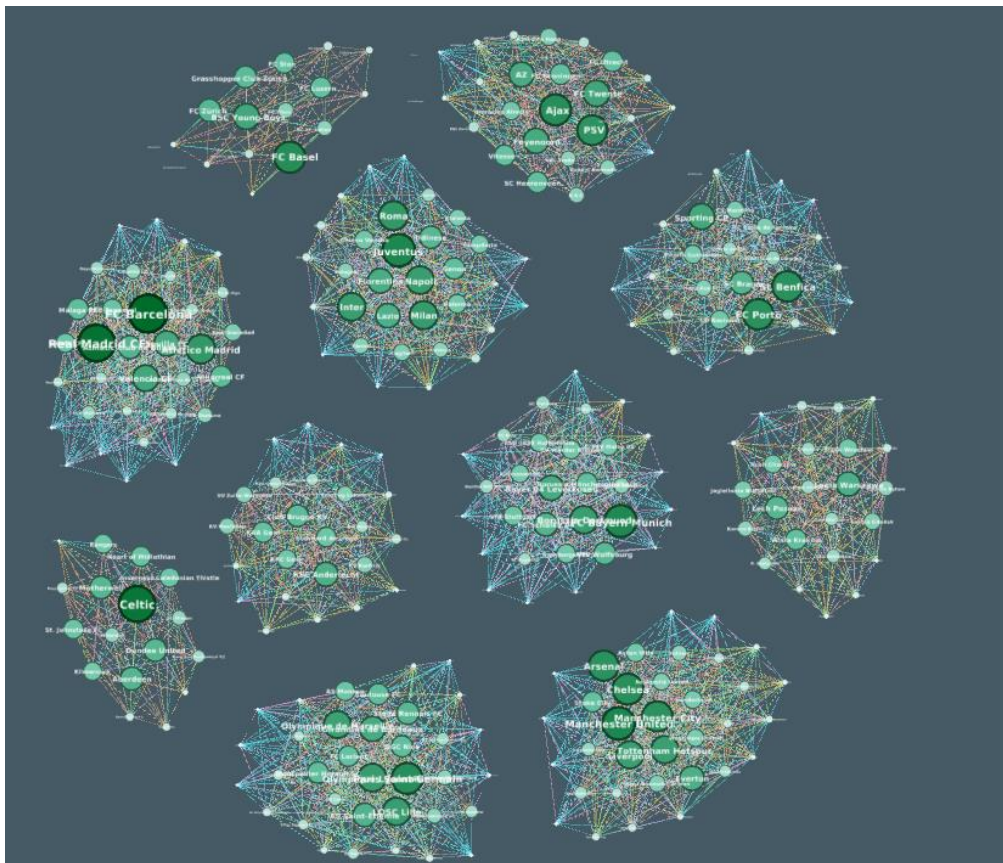


Ilustración 10: Visualización con Gephi de las diferentes ligas

El grafo separa nuestros datos por liga, ya que no hay enfrentamientos entre equipos de ligas diferentes. Se puede analizar viendo los diferentes grupos, lo desigual que es una liga, como por ejemplo la escocesa, donde el Celtic es el nodo más grande con diferencia, o lo igualada que está, teniendo de ejemplo el caso de la Premier League, donde hay al menos 4 equipos con un número de puntos parecido (tamaño similar).

Teniendo los anteriores análisis, pasamos a un proceso de feature selection, donde tendremos que buscar los mejores parámetros para hacer una predicción. Parece que lo más lógico es utilizar la fuerza de los equipos, es decir, los ratings de sus jugadores.

Para ello, agregamos los ratings de los 11 jugadores sin ponderar, es decir, dándole la misma importancia a cualquier jugador del equipo. Si visualizamos los partidos, siendo el eje X la fuerza del equipo local y el eje Y la fuerza del equipo visitante, podemos observar ciertos patrones. Para visualizar otra dimensión más

coloreamos los puntos según cada símbolo de la quiniela (1, X, 2), es decir, ganador el equipo local, equipo visitante o empate.

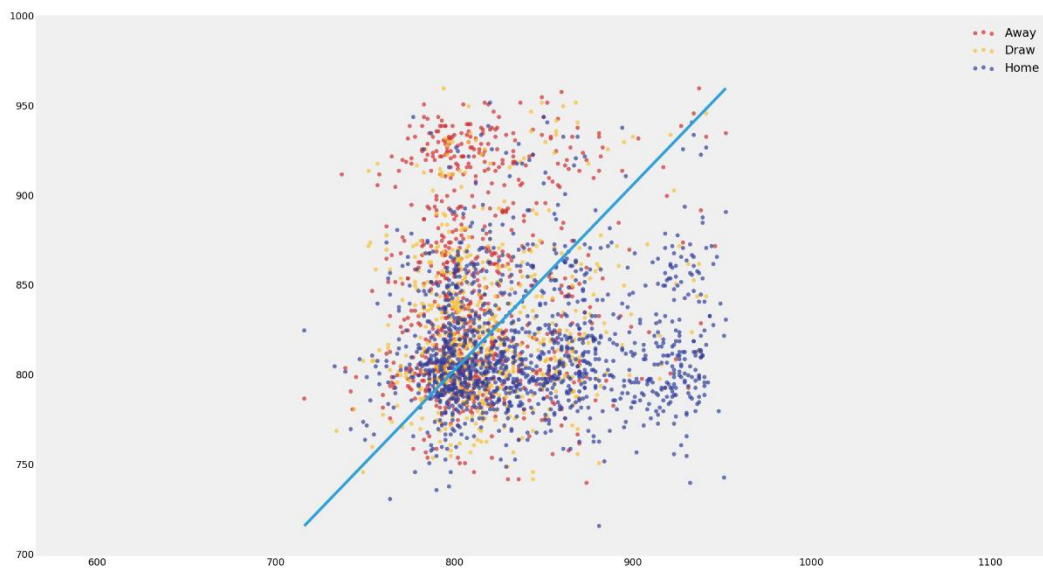


Ilustración 11: Visualización de los partidos de la liga española dependiendo de los ratings de ambos equipos

Se puede ver como hay una mayor tendencia del equipo fuerte, el que cuenta con un rating mayor, a ganar, aunque es mucho más claro cuando este equipo juega de local que de visitante. El gráfico también nos muestra la aleatoriedad y, por tanto, mayor dificultad de predicción, de los empates. Es extrapolable al comportamiento de todas las ligas, pero utilizamos el de la liga española porque de lo contrario no sería posible entender todas estas cuestiones con la visualización, ya que al haber tantos partidos los puntos se superponen y nos complica mucho dicha tarea.

También, aprovechando que nuestra base de datos nos proporciona las cuotas de varias casas de apuestas, las utilizaremos como variables de entrada para los modelos. Con esto intentaremos aumentar la información para poder hacer mejores predicciones.

Calculamos y visualizamos la matriz de covarianza para ver las relaciones entre las diferentes variables que hemos elegido: rating equipo local, rating equipo

visitante, cuota local, cuota empate y cuota visitante. La cuota no es más que la inversa de la probabilidad asociada a que se de ese resultado.

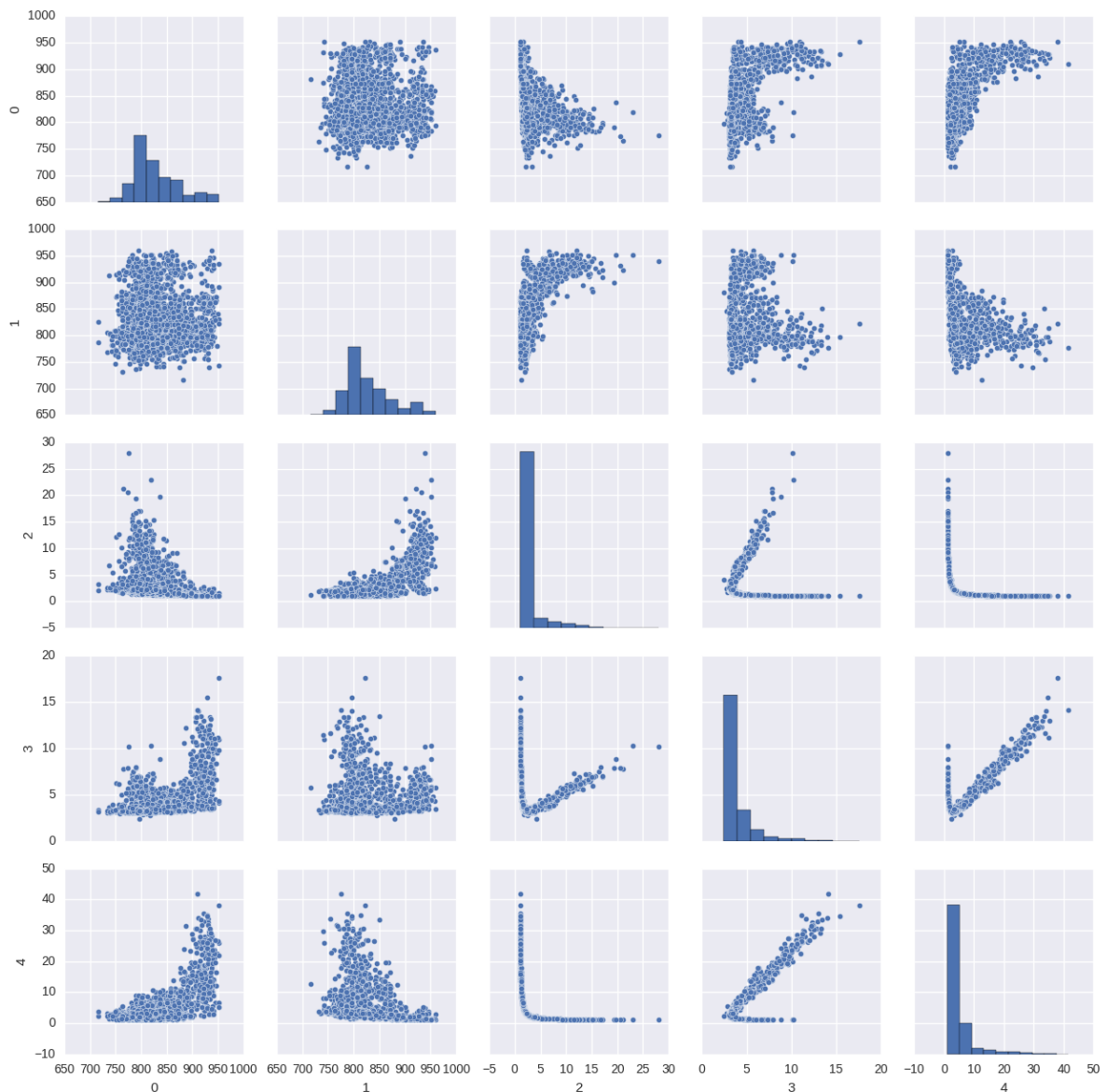


Ilustración 12: Matriz de diagramas de dispersión de todos los pares de variables e histograma de cada variable

Las variables que se tienen en cuenta en la anterior ilustración son los ratings del equipo local (variable 0), ratings del equipo visitante (variable 1) y las cuotas de las casas de apuestas, gana equipo local (variable 2), empatan (variable 3) y gana equipo visitante (variable 4). La matriz enfrenta a cada par de variables para conocer la correlación entre ellas, además de dibujar el histograma de cada variable en la diagonal de la misma, haciendo que podamos sacar conclusiones de las características que hemos elegido en nuestro problema.

Visualizando esta matriz de diagramas de dispersión, se pueden observar patrones en los datos, sobre todo en la parte de las cuotas, pero también en los partidos frente a las cuotas. Para visualizar en una gráfica todo esto elegimos una forma personal de hacerlo, siendo el eje X la fuerza de los equipos y el eje Y las cuotas. Los dos ejes están normalizados para una mejor visualización, ya que los valores de las variables no están en los mismos rangos.

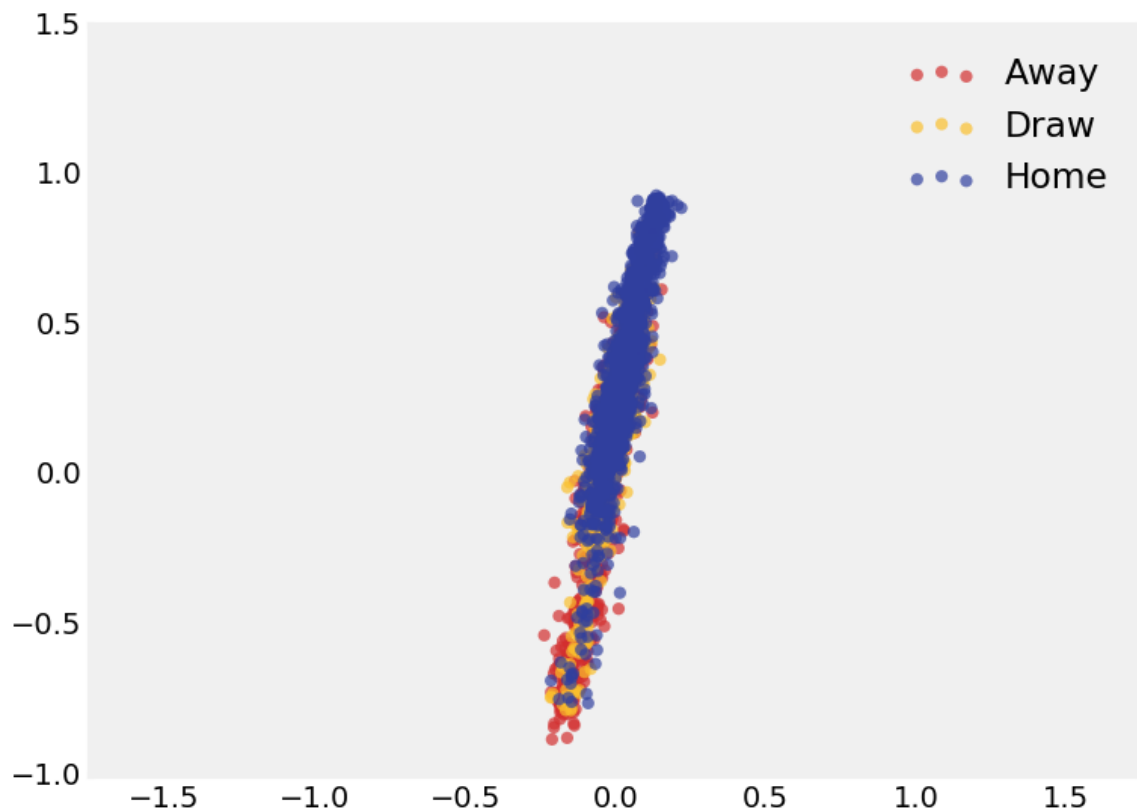


Ilustración 13: Visualización de los resultados a través de los ratings de los equipos y de las cuotas de las casas de apuestas

Una vez más, se confirman los patrones que ya habíamos visto antes, el equipo local suele ganar, además de ser más probable que lo haga el equipo fuerte, y las cuotas pueden aportar robustez al modelo final.

5.3 Machine Learning

Una máquina puede hacer una tarea sin haber sido expresamente programada para ello. Como si de un humano se tratase, lo único que necesita son datos, ejemplos de aquello que se quiere aprender, con la diferencia de que los seres

humanos tenemos mucha menos capacidad de procesamiento que una máquina, la cual puede hacer millones de cálculos por segundo.

El Machine Learning está dentro del gran campo de la inteligencia artificial, centrándose en que las máquinas puedan aprender de los problemas para dar unas mejores soluciones a ellos. Esto se consigue aplicando técnicas estadísticas desde una simple regresión lineal a una red neuronal, donde se modela el comportamiento del cerebro humano, a los datos de entrada al sistema.

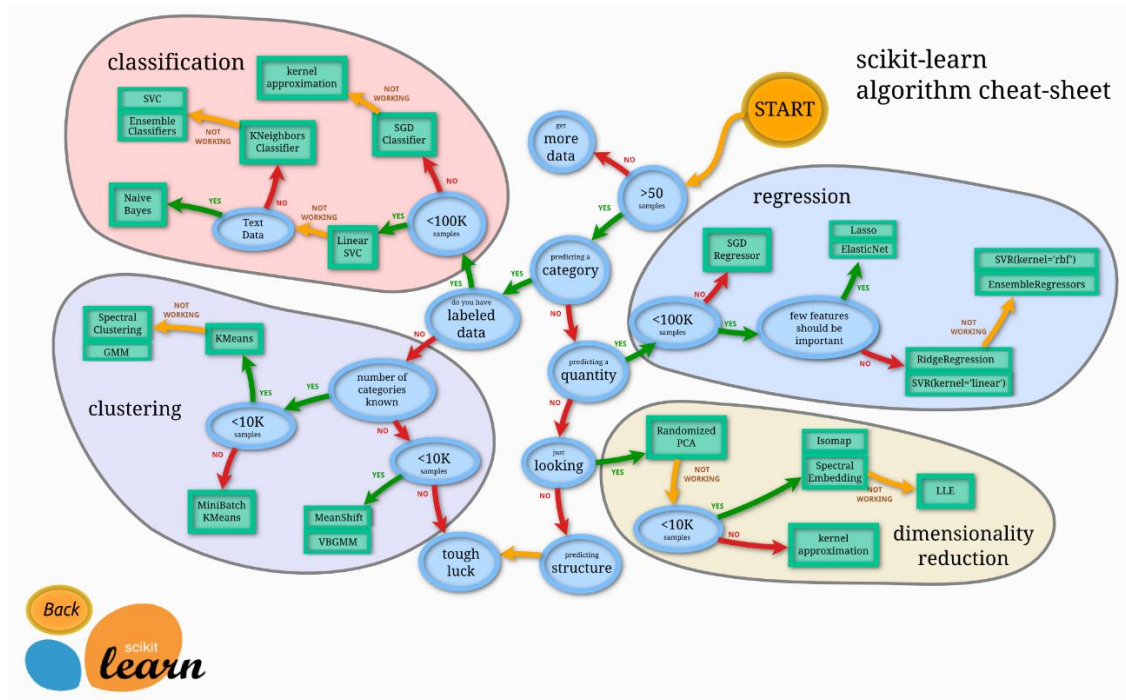


Ilustración 14: Cheat sheet de scikit-learn [24]

Existen muchos algoritmos de Machine Learning en la actualidad y en este proyecto estudiaremos muchos de los más conocidos. Con las variables elegidas y los patrones observados, procedemos a aplicar las diferentes técnicas de Machine Learning que elegimos en nuestro estudio a nuestros datos.

Dentro del Machine Learning se distinguen tres grandes ramas:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje por refuerzo

En este proyecto se va a aplicar las dos primeras formas de aprendizaje máquina, ya que el aprendizaje por refuerzo requeriría otro tipo de datos de los que no disponemos.

Nuestro conjunto de datos y las variables que hemos elegido encajan dentro del aprendizaje supervisado, ya que todos los ejemplos están etiquetados. Además, se trata de un problema de clasificación, ya que está planteado como un predictor de tipo quiniela (1, X, 2), por tanto, cuenta con tres categorías, en lo que podría dominarse como un problema de clasificación multiclase y son el tipo de algoritmos que más vamos a utilizar.

En cuanto a la forma en la que se va a aplicar el aprendizaje no supervisado a nuestro problema va a ser eliminando las etiquetas de todos los datos e intentar encontrar grupos de datos similares en ellos mediante técnicas de clustering. En el apartado de aprendizaje no supervisado profundizaremos en la forma en la que nos puede ayudar el clustering a mejorar las prestaciones de nuestros modelos.

5.4 Aprendizaje supervisado

Estas técnicas consisten en darle los datos etiquetados a los algoritmos, es decir, ejemplos donde se saben los valores de las variables que hemos elegido y el resultado final. Por tanto, el modelo aprenderá de estos ejemplos y podrá predecir los valores de salida, el resultado de un partido, para nuevos datos.

Tenemos las variables de entrada X , siendo las que hemos elegido anteriormente, rating equipo local, rating equipo visitante, cuota victoria local, cuota victoria visitante y cuota empate, y la salida Y , que en nuestro caso es resultado final del partido en forma de signo de la quiniela, es decir 1, X, 2.

Dentro del aprendizaje supervisado, existen varios tipos de problemas y estudiaremos como aplicarlos a nuestros datos para conseguir una buena

predicción, siempre teniendo en cuenta lo complicado que es predecir el fútbol, siendo uno de los deportes más impredecibles que existen.

El aprendizaje supervisado se puede dividir en tareas de clasificación y tareas de regresión. La primera consiste en hacer una predicción sobre ciertos datos, donde la salida es una categoría y las tareas de regresión se aplican para las predicciones de valores continuos. Parece que nuestro problema, tal y como lo hemos planteado, se trata de un problema de clasificación al ser la salida el signo de la quiniela.

Aunque vamos a aplicar algoritmos de los dos tipos de problemas que acabamos de mencionar, empezaremos con los de clasificación, ya que son los que más se ajustan a nuestra forma de representar las variables de entrada y salida, los signos de la quiniela son las categorías que queremos predecir.

Antes de empezar con los algoritmos es necesario introducir el sobreajuste y la sobregeneralización, o con sus términos en inglés, overfitting y underfitting. El sobreajuste es cuando un modelo se ajusta demasiado bien a los datos de entrenamiento, pero generaliza mal cuando lleguen nuevos datos desconocidos. Por el contrario, la sobregeneralización, que es cuando el modelo comete un error al no tener en cuenta ciertos patrones que son importantes para poder generalizar bien a nuevos datos y que funcione bien con los datos de entrenamiento.

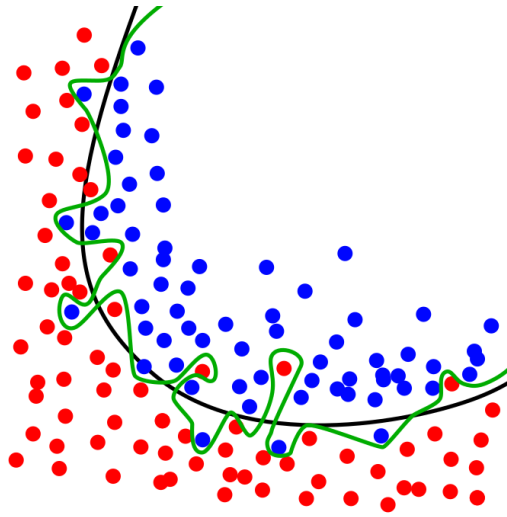


Ilustración 15: Ejemplo de sobreajuste

Estrechamente relacionado con los problemas de ajuste, tenemos que evitar que nuestro algoritmo tenga una varianza alta, lo que causará overfitting, así como evitar tener un alto sesgo, ya que produciría overfitting. Para saber si nos enfrentamos ante un problema de varianza o de sesgo tendremos que fijarnos en el error cuando comparamos las predicciones con los datos de entrenamiento y en el error de la validación cruzada, es decir, separar los datos de prueba para enseñarle al algoritmo con una parte de ellos y probar las predicciones con el resto, los cuales ya están etiquetados y por tanto sabemos el resultado.

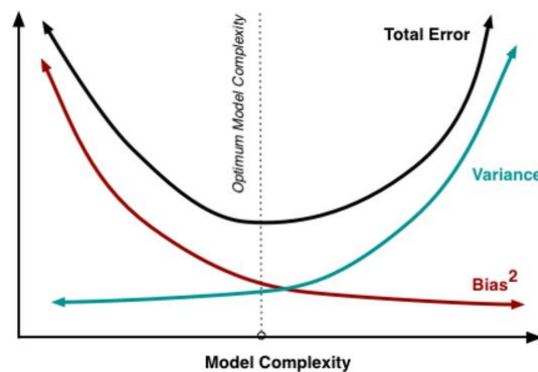


Ilustración 16: Compensación entre sesgo y varianza [25]

Si el error de entrenamiento y el de la validación cruzada son altos y parecidos entre sí, estaremos ante una solución con un alto sesgo y sobregeneralización. Si, en cambio, el error para los datos de prueba es muy bajo, incluso 0, y el error de

la validación cruzada es alto, tendremos alta varianza y, por tanto, un problema de sobreajuste.

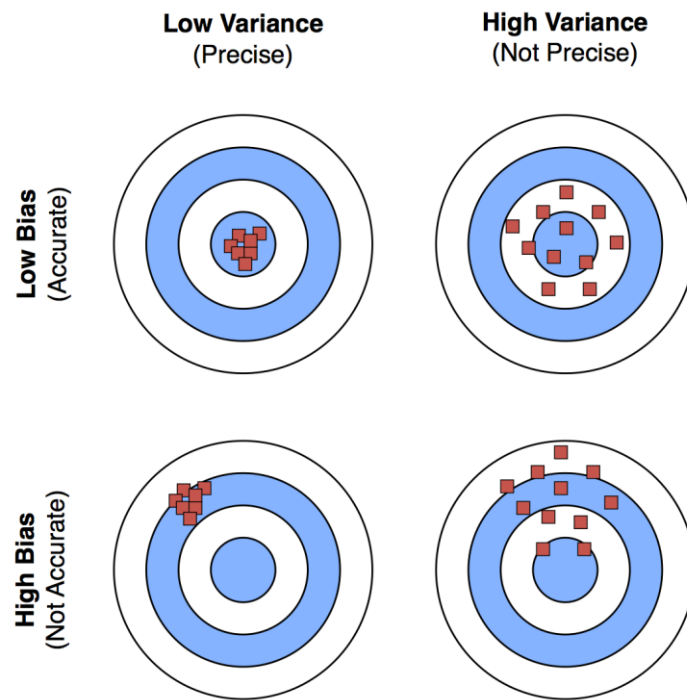


Ilustración 17: Precisión frente a exactitud

En los siguientes apartados se presentará una pequeña introducción de los algoritmos utilizados con sus pros y contras, para posteriormente, explicar los parámetros utilizados para su fácil reproducción, así como la forma de medir la precisión de los diferentes algoritmos,

5.5 Clasificación

Para enfocar nuestros datos a un problema de clasificación tenemos que representar la salida como diferentes categorías, por lo que utilizaremos el famoso 1, X, 2 de la quiniela mencionado anteriormente.

Aplicaremos los siguientes algoritmos de clasificación para nuestro estudio y los compararemos a través de ciertas métricas para conocer cuál de ellos rinde mejor:

- Decision Tree
- SVM (Support Vector Machine)

- KNN (K-Nearest Neighbors)

5.5.1 Decision Tree

Como para todos los algoritmos que vamos a aplicar en el proyecto nos apoyaremos de scikit-learn [26], ya que cuenta con una cantidad enorme de algoritmos de Machine Learning y una muy buena documentación con su explicación y uso de los métodos.

Un árbol de decisión como clasificador consiste en encontrar delimitadores dentro de los datos para ir tomando decisiones. Es capaz de ver relaciones no lineales en los datos, además de ser muy facil de entender e interpretar, lo que hacen que este algoritmo sea uno de los más utilizados dentro del aprendizaje supervisado.

Los arboles de decisión son modelos estadísticos que funcionan con bloques si-entonces, donde si se cumple una cierta condición será más probable que se acabe clasificando una u otra categoría. Encontrar la condición es una tarea de exploración, donde tienen que encontrarse límites en los datos que sean óptimos, los cuales no siempre son triviales.

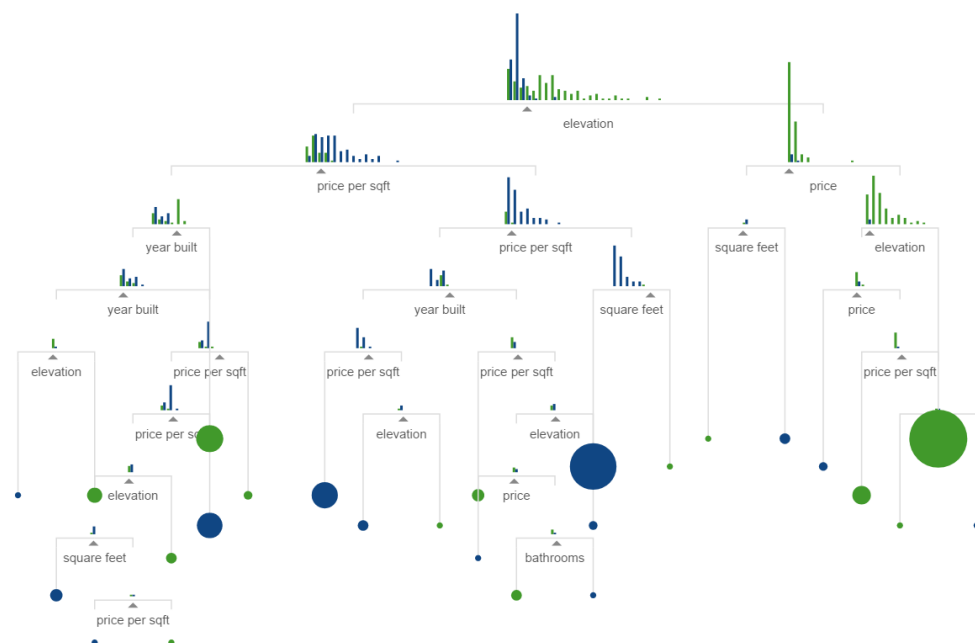


Ilustración 18: Ejemplo de árbol de decisión aplicado a clasificación de viviendas en Nueva York o San Francisco [27]

El problema que tienen los árboles de decisión es que suelen sobreajustar. Es fácil que lo hagan ya que encuentran límites muy específicos en los datos. Si no elegimos la profundidad adecuada tendremos un problema de este tipo y aunque en las pruebas nos salga una precisión muy alta no va a generalizar bien para nuevos datos.

5.5.2 SVM

Scikit-learn nos proporciona una manera muy sencilla para poder aplicar una máquina de soporte vectorial a nuestros datos [28]. Este algoritmo de clasificación construye una división en forma de línea, plano o hiperplano, según la dimensionalidad del problema, donde separa cada clase de las demás. Para esta labor, proyecta los datos a un espacio donde sea posible separarlos a través de dicha línea o similar, donde solo tiene en cuenta las fronteras de cada conjunto etiquetado de datos, los cuales son denominados vectores de soporte.

Para predecir los valores futuros comprueba en que subespacio se encuentra la muestra en cuestión y se le asigna su etiqueta. Las máquinas de soporte vectorial clasifican según su kernel, el cual es una función con la que dividir los datos. Puede ser lineal, cuadrático, gaussiano, e incluso un árbol de decisión.

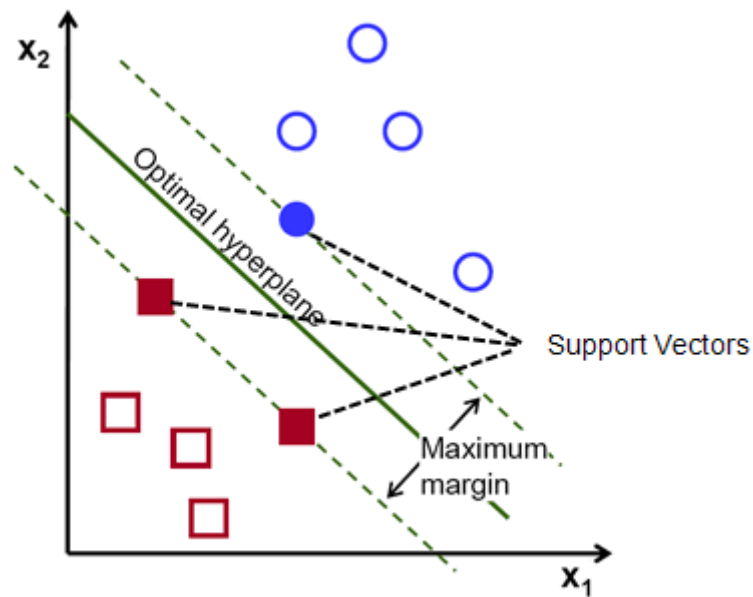


Ilustración 19: Ejemplo de SVM [29]

Una de las ventajas de utilizar este método es que no requiere una cantidad enorme de datos para funcionar bien, lo que hace que no sea tan problemático tener las limitaciones de nuestro set de datos.

5.5.3 KNN

K-Nearest Neighbors es también uno de los algoritmos más sencillos que se aplican en el aprendizaje máquina, pero a la vez uno de los más utilizados y que mejor funcionan.

Lo implementaremos con scikit-learn [30] y aplicaremos a nuestros datos. Lo que hace KNN es, para un nuevo dato calcular los k puntos más cercanos y ver que clase domina entre esos vecinos. Para la clase que sea dominante, KNN clasificará el nuevo dato como perteneciente a ella.

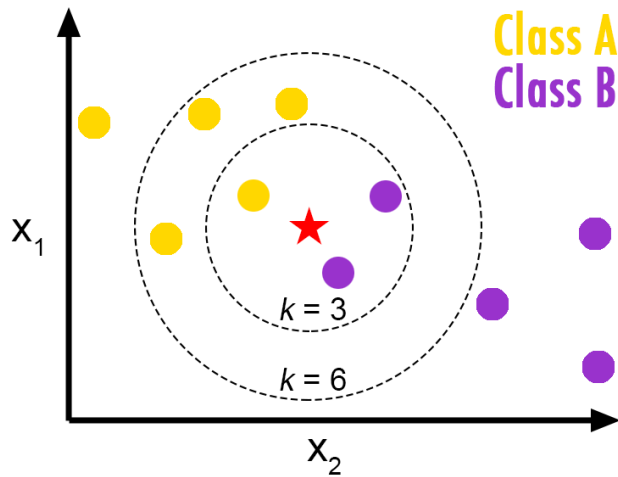


Ilustración 20: Ejemplo de KNN [31]

Se puede ver en la ilustración como para diferentes valores de k el algoritmo puede hacer decisiones completamente diferentes. Aumentar el valor de k suele mejorar la decisión frente a ruido, aunque hay que tener cuidado con elegir un número muy alto de vecinos, ya que puede que haya clases desbalanceadas y entonces errará con la predicción final, además de que puede no generalizar bien a nuevos datos.

5.5.4 Logistic Regression

La regresión logística, pese a que su nombre parece indicar lo contrario, pertenece al grupo de clasificadores.

Al igual que en el caso que los demás algoritmos, scikit-learn nos proporcionará el esqueleto para poder implementar este algoritmo a nuestros datos. El objeto encargado de ello es `linear_model.LogisticRegression` [32], que nos brinda los métodos necesarios para poder aplicar una regresión logística a nuestros datos de entrada.

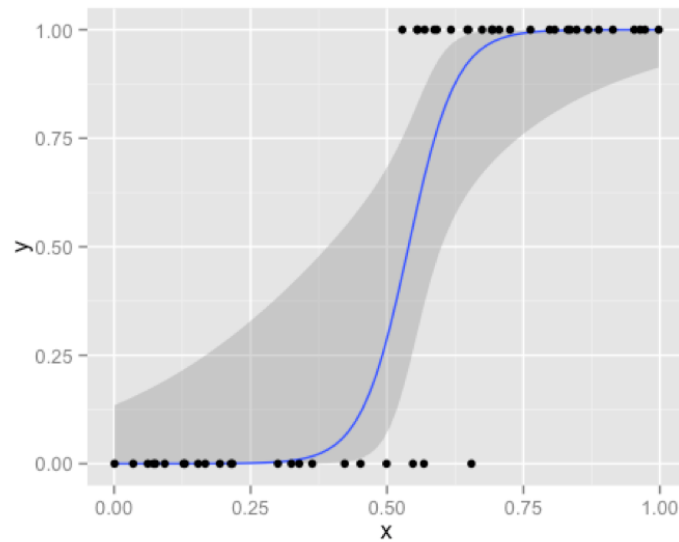


Ilustración 21: Ejemplo de regresión logística [33]

Las ventajas de este algoritmo es que es capaz de ver probabilidades asociadas a que ocurran eventos, es decir, tendrá en cuenta la probabilidad de que haya una diferencia de goles determinada, lo que nos permitirá realizar una predicción.

5.6 Regresión

A diferencia de los problemas de clasificación, en la regresión no existen categorías, sino valores continuos, por lo que tendremos que transformar nuestras categorías en un valor numérico. Para esta nueva variable de salida se utilizará la diferencia de goles entre el equipo local y el visitante, resultando un valor negativo en caso de que gane el equipo visitante y positivo si gana el visitante (0 en caso de empate).

Para el caso de predecir un empate, tendremos que poner un umbral en el resultado del predictor, ya que carece de sentido decidir que dos equipos van a empatar solo si la salida es un 0 (ambos equipos meten los mismos goles). Este umbral debería estar en ± 1 , ya que si la diferencia entre un equipo y otro es menor a 1 gol significaría que han quedado en empate.

En este apartado analizaremos tan solo el algoritmo de regresión lineal, ya que, como se ha indicado en apartados anteriores, nuestro problema está enfocado a la clasificación.

5.6.1 Linear Regression

Scikit-learn nos ofrece con `linear_model.LinearRegression` [34] una forma muy sencilla de implementar un modelo de regresión lineal a nuestros datos. Con este algoritmo se busca encontrar relaciones lineales entre los datos, lo cual, aunque parezca algo demasiado simple, se puede aplicar debido a que las variables de entrada que utilizamos para los modelos son también muy simples.

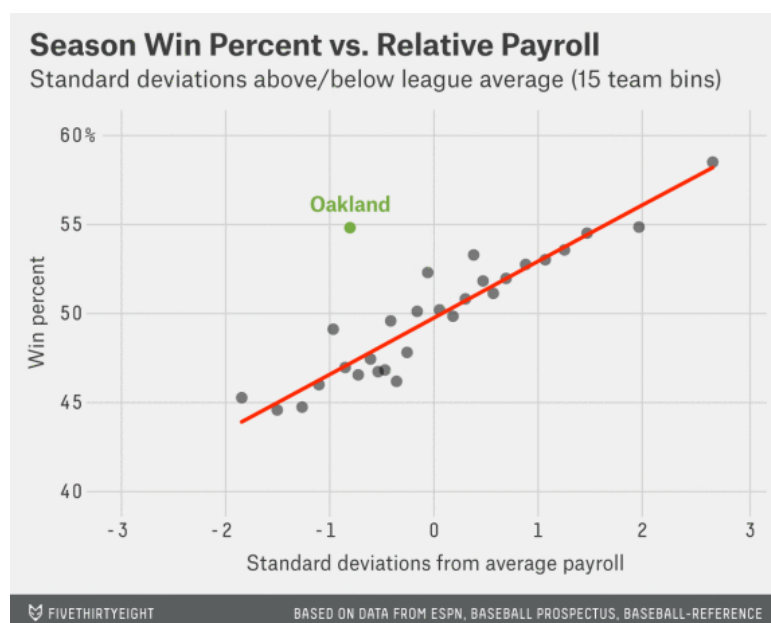


Ilustración 22: Ejemplo de regresión lineal aplicado al famoso caso de moneyball [35]

Para medir la predicción de este algoritmo tendremos en cuenta si ha acertado el signo de la quiniela, es decir, aunque sea capaz de devolver la diferencia de goles entre equipo local y visitante solo tendremos en cuenta que equipo gana y lo compararemos con los ejemplos que tenemos.

5.7 Aprendizaje no supervisado

En el aprendizaje no supervisado, contrariamente a lo mencionado en capítulos anteriores, donde todos los algoritmos conocían ejemplos etiquetados de los

datos y aprendían de ellos, no se conoce la variable de salida de los datos de entrenamiento. Este tipo de aprendizaje es muy común ya que muchos de los conjuntos de datos existentes no contienen la etiqueta asociada a cada ejemplo.

Para nuestro estudio el algoritmo tendrá que realizar una predicción conociendo la calidad de ambos equipos, a través de los ratings, y las cuotas de las casas de apuestas, pero sin tener ni un solo ejemplo que nos relacione la variable que queremos predecir (resultado de un partido de fútbol), con las variables de entrada.

Dentro de este tipo de aprendizaje existen muchos algoritmos, aunque nos centraremos en los de clustering, los cuales agrupan los datos según patrones comunes. Veremos cómo agrupar nuestros datos y si tienen o no sentido esos grupos, centrándonos en los dos tipos de clustering, por centroides o por densidad, profundizando más adelante en ambos tipos de clustering.

El desarrollo de este capítulo será diferente al anterior, ya que se trata de un estudio diferente a lo planteado para el aprendizaje supervisado. Trataremos de comprobar en qué puede ayudar el clustering a la mejora de prestaciones, basándonos en un paper sobre la utilidad del clustering en la mejora de prestaciones para las tareas de predicción [9].

5.7.1 Clustering para mejora de prestaciones

Dividiremos este apartado en dos modelos claramente diferenciados, uno con las mismas variables que venimos utilizando anteriormente y un segundo modelo, esta vez un poco más complejo, para poder aplicar ciertas técnicas populares en el aprendizaje no supervisado como la reducción de dimensionalidad.

Para el modelo más simple, realizaremos un clustering por instancias, es decir, buscando grupos en los datos de entrada, para poder visualizar estos grupos y sacar conclusiones de ellos. Este primero modelo buscará tipos de partidos que compartan características parecidas, lo que parece bastante intuitivo, ya que no

es lo mismo predecir un partido entre dos grandes equipos que entre dos pequeños. Para el segundo modelo, podremos aplicar las predicciones después de haber hecho el clustering y comprobar si puede ayudar, o no, a la mejora de prestaciones.

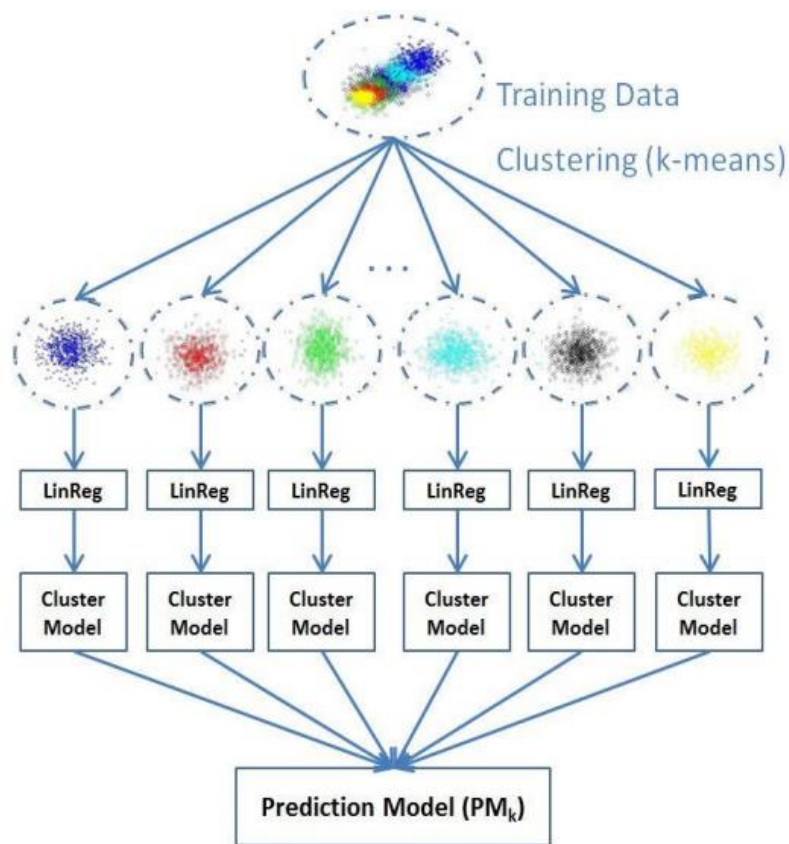


Ilustración 23: Ejemplo de aplicación del estudio propuesto

En cambio, para el segundo modelo, aplicaremos un clustering por variables, dada su matriz de covarianza, para poder distinguir entre los datos que sigan correlaciones con la variable de salida similares. A continuación podremos aplicar algoritmos de reducción de dimensionalidad y selección de características, para finalmente realizar una predicción y comprobar si el clustering puede ayudar a la mejora de prestaciones de nuestros modelos anteriores.

5.7.2 Modelo simple

Primero aplicaremos los algoritmos de clustering elegidos al modelo más simple de los dos.

5.7.2.1 K-means

K-means es el algoritmo de clustering más utilizado del mundo, siendo su simpleza y facilidad de entendimiento dos de las características que más le ayudan a ello. Utilizaremos `sklearn.cluster.KMeans` [36] para aplicar este algoritmo de clustering a nuestros datos sin etiquetar.

Para su implementación tan solo tendremos que tener en cuenta el parámetro `k`, el cual se refiere al número de clusters en los que queremos dividir nuestros datos. Esto convierte a K-means en un algoritmo que funciona mejor cuando los datos son fáciles de visualizar y podemos distinguir ciertos grupos en ellos, aunque hay formas de calcular el valor de `k` óptimo sin tener que elegirlo por intuición.

El funcionamiento de k-means es muy sencillo:

- Elegimos `k` para decirle al algoritmo en cuantos clusters queremos que nos divida los datos. Comienza colocando `k` centroids al azar (centros de cluster) y luego resuelve de forma iterativa hasta converger lo siguiente:
 - Asigna cada punto (dato) a su centroide más cercano, haciéndole así miembro de ese cluster.
 - Calcula la nueva posición del centroide, calculando la posición media de todos los puntos que pertenecen a ese cluster.
 - Volver al paso 1 si no ha convergido (los puntos no cambian ni tampoco los centroides).

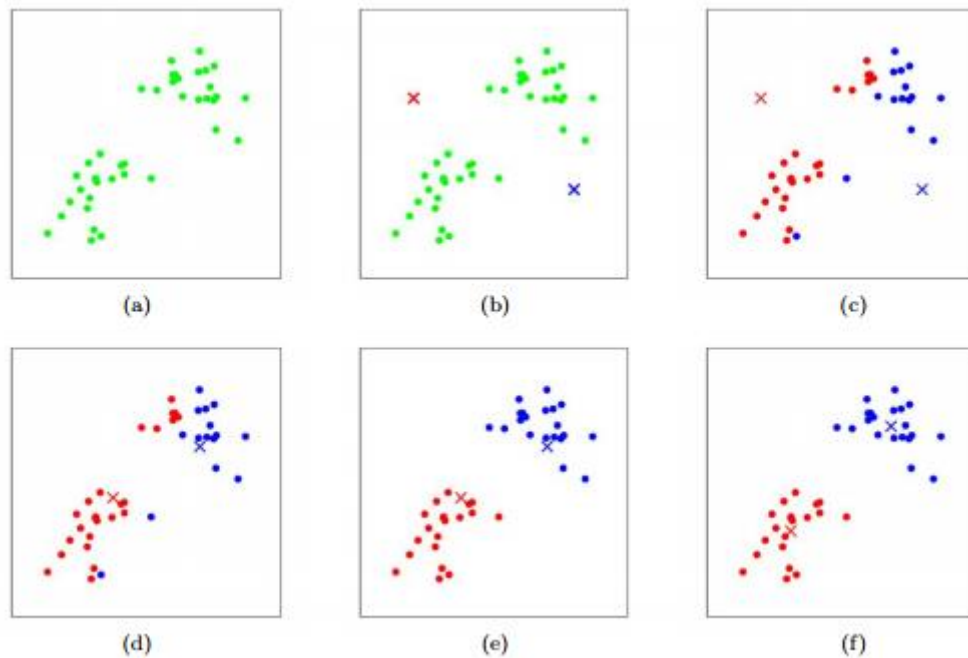


Ilustración 24: Visualización de los pasos que sigue k-means para encontrar clusters en los datos [37]

El objeto de KMeans se ha inicializado con parámetros:

- `random_state = 0`, Para que la semilla sea siempre la misma y sea más fácil hacer pruebas.
- `init = 'k-means++'`, hace que el algoritmo converja más rápido.
- `n_clusters=5`, de forma intuitiva después de realizar pruebas
- Los demás parámetros tendrán sus valores por defecto

Para visualizar los resultados vamos a enfrentar en los ejes de coordenadas los ratings de ambos equipos, siendo el eje X los ratings agregados del equipo local y el eje Y los del equipo visitante.

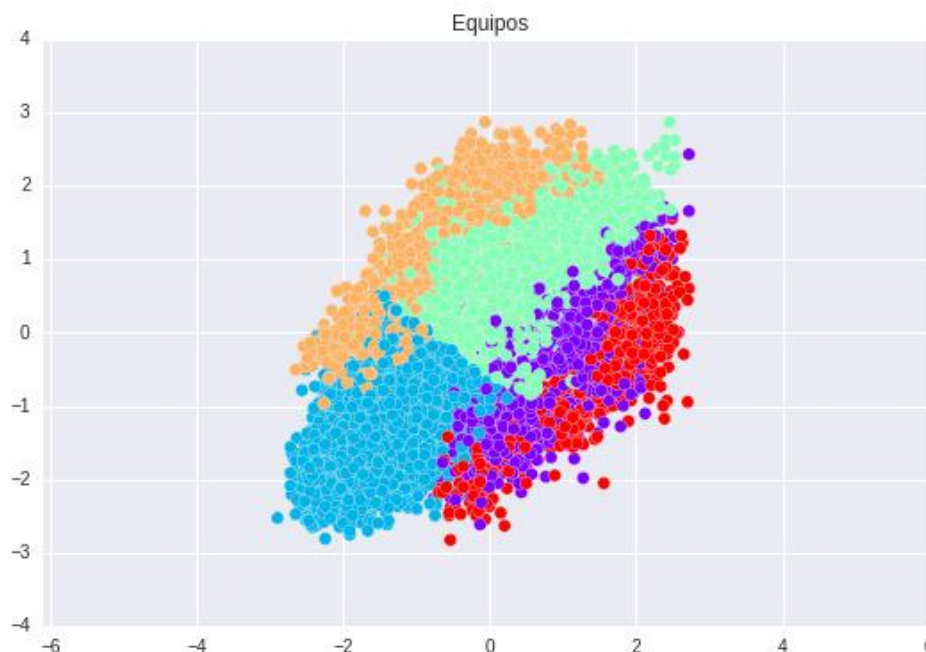


Ilustración 25: Clusters de k-means (colores), visualizado por calidad de los equipos

Viendo los colores podemos observar cómo se distribuyen los 5 grupos, viendo como los partidos que están entre los equipos medianos y los limitados forman un gran grupo, de la misma forma que lo hacen los partidos entre equipos que van desde medianos a fuertes (nótese que los ejes tienen valores escalados).

Los colores naranja y rojo parecen tener cierto sentido, ya que se tratan de partidos en los que el equipo local, en caso del grupo rojo, o equipo visitante, caso del grupo naranja, son bastante superiores a su rival. En cambio, superponiéndose en muchas zonas al rojo pero más cercano a la recta de pendiente 1, donde estarían enfrentados los equipos con un mismo rating, aparece un grupo violeta que no parece tener mucho sentido todavía.

Para ver porque ha podido agrupar de esa forma, visualizamos también respecto a las cuotas de victoria, eje de abscisas, y derrota, eje de ordenadas, preescaladas para poder aplicar después los algoritmos de predicción correctamente, ya que los órdenes de magnitud no son iguales para los ratings que para las cuotas.

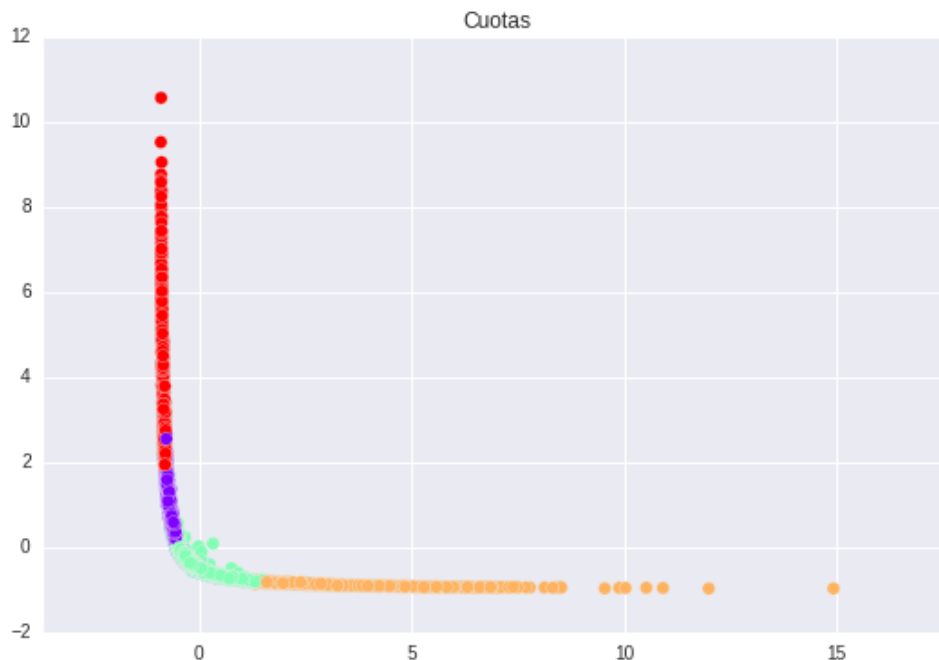


Ilustración 26: Clusters de k-means (colores), visualizado por cuotas de las casas de apuestas

Parece ser que el violeta toma sentido, ya que se debe a partidos donde los equipos locales eran más fuertes pero las cuotas favorecían al equipo visitante, tratándose de un comportamiento diferente a lo esperado, suficiente como para crear un grupo nuevo.

En esta ocasión no visualizaremos las 5 variables a la vez como hicimos en el apartado 1, ya que los puntos están tan pegados que los 5 grupos se ven muy juntos y engaña mucho, ya que matplotlib pinta encima los puntos que se colocan por último lugar en el código, como le pasa al grupo azul en la visualización de las cuotas, donde parece que no existe.

5.7.2.2 DBSCAN

K-means no es siempre la mejor opción para el clustering, sobre todo cuando los datos dibujan ciertas formas, y una de las alternativas son los algoritmos de clustering basados en densidades. Los algoritmos que se basan en la densidad tienen como premisa que todos los datos están dibujados con una función de densidad de probabilidad, por lo que utilizar una estimación de esa función debería darnos muy buenos resultados.

Comparado con k-means, un algoritmo teórico basado en densidades tiene diferentes pros y contras:

Pros

- Podemos detectar formas mucho más complejas en los datos.
- No necesitamos saber k (número de clusters).
- Automaticamente encontramos los outliers.

Contras

- Requiere una función de distancia.
- No es tan escalable como k-means.

Como entrada tiene epsilon, que mide la distancia máxima en la que se buscarán vecinos y min_samples, variable que nos dice el mínimo número de vecinos para considerar a un punto dentro de un cluster, dichos puntos con el número mínimo de vecinos serán los puntos de frontera. Para nuestra implementación con cluster.DBSCAN [38].

El algoritmo busca los core points, o puntos de núcleo, que son los puntos que tienen la mayor densidad, están rodeados de una mayor cantidad de puntos y serán el centro de los clusters. Visita todos los puntos hasta llegar a los puntos de frontera y luego pasa a otro cluster. Los outliers son considerados como ruido y no los categoriza en ningún cluster.

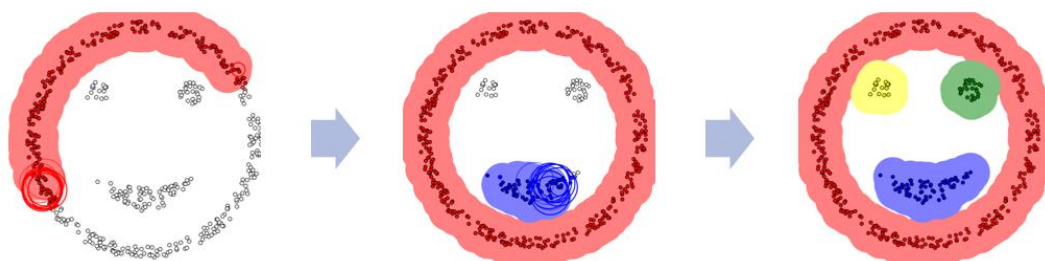


Ilustración 27: Visualización del algoritmo DBSCAN encontrando los grupos por densidades

DBSCAN es capaz de distinguir entre 9 grupos de datos dentro de nuestro problema, aunque realmente no parece que para este primer modelo vaya a tener

mucho sentido aplicarlo, ya que los grupos no parecen seguir una distribución lógica, como vimos que si pasaba en k-means.

Como parámetros de entrada utilizaremos (ambos después de realizar varias pruebas):

- $\text{eps}=0.3$
- $\text{min_samples}=10$



Ilustración 28: Visualización de los grupos que forma DBSCAN por equipos

Como nuestros datos no contienen una estructura compleja, sino que son una sola masa, a DBSCAN le resulta muy complicado obtener los grupos de partidos que guardan una cierta relación. Lo poco que se podría sacar como conclusión de esta visualización sería el hecho de que se forman más grupos del espacio donde es superior el equipo visitante al local, quizás porque es donde hay más sorpresas gracias al factor campo.

5.7.3 Modelo desagregado

En este modelo ampliaremos el uso de características, para poder aplicar el clustering a la matriz de covarianza y ver en que grupos se dividen las variables y

hacer la selección. Veremos como de buena era nuestra intuición de sumar todos los ratings globales de un jugador y de hacer una media de las cuotas de las casas de apuestas.

5.7.3.1 Extracción y tratamiento de la información para cada partido

Esta vez utilizaremos más columnas para nuestra matriz de partidos. Adentrándonos en la base de datos vemos que podemos utilizar atributos de cada equipo, así como ratings de los jugadores en facetas del juego más concretas. En vez del rating global utilizar puntuación de regate, definición o pase, por ejemplo.

Nos encontramos con un problema en nuestro set de datos, a la tabla de Team_Attributes le falta mucha información de BuildUpDribbling, más de un 66%, lo que nos limitaría muchísimo el número de partidos que tenemos (tendríamos que eliminar el 34% restante de partidos para no tratar con datos nulos). La solución podría ser rellenar los huecos con una media del resto de las características, predecir esos valores con una regresión o similar y, por último, ignorar esa columna completa. Como hacer una media no parece lo más razonable y predecir el 66% de los valores con el 34% restante tampoco, decidimos eliminar dicha columna.

El segundo problema viene al intentar mapear cada atributo de los equipos (misma tabla que en el apartado anterior), ya que al desarrollar un pequeño código de Python para mapear cada atributo a su correspondiente equipo por partido, vemos que hay muchos que tienen una fecha posterior a la fecha del mismo, por lo que tampoco podríamos utilizarlo. Si fuera un problema poco común bastaría con eliminar los partidos en los que ocurre, pero son más de un 75% de los partidos, totalmente inviable.

Por tanto, la información en conjunto de los equipos sacrifica demasiado nuestro número de partidos del dataset y no parece que sea una opción viable. No

podremos alimentar a nuestro modelo con lo ofensivo o defensivo que es un equipo, quizás en una futura versión de la base de datos sea posible.

Atacamos entonces el apartado de jugadores, donde tenemos problemas similares al de BuildUpDribbling con varias columnas:

- sliding_tackle
- vision
- jumping
- balance
- agility
- curve
- volleys
- attacking_work_rate

Por tanto, estos atributos los desechemos desde un principio, ya que nos quita un porcentaje de nuestro dataset demasiado elevado. Una vez eliminadas esas características, tenemos para elegir entre una larga lista de variables:

- height
- weight
- overall_rating
- potential
- crossing
- finishing
- heading_accuracy
- short_passing
- dribbling
- free_kick_accuracy
- long_passing
- ball_control
- acceleration
- sprint_speed

- reactions
- shot_power
- stamina
- strength
- aggression
- interceptions
- positioning
- penalties
- marking
- standing_tackle
- gk_diving
- gk_handling
- gk_kicking
- gk_positioning
- gk_reflexes

Cogerlas todas podría parecer lo más lógico, pero el modelo tendría 638 (29 variables por 22 jugadores) features solo con la información de los jugadores. Un modelo quizás demasiado complicado para la tarea que estamos intentando estudiar, pero que quizás en un estudio futuro con técnicas de Deep Learning puede ser muy interesante.

Finalmente vamos a utilizar las mismas variables del modelo 1 pero desagregándolas, es decir, un rating para cada jugador y las 6 casas de apuestas con sus respectivas cuotas cada una (tendrán una gran correlación). Gracias a esta simple expansión tenemos ya disponible un vector X de 40 variables con las que, ahora sí, tiene sentido aplicar nuestro estudio.

Al utilizar las mismas variables que en el modelo 1 pero extendidas podremos, además, medir como de buena era nuestra intuición de reducción de la dimensionalidad.

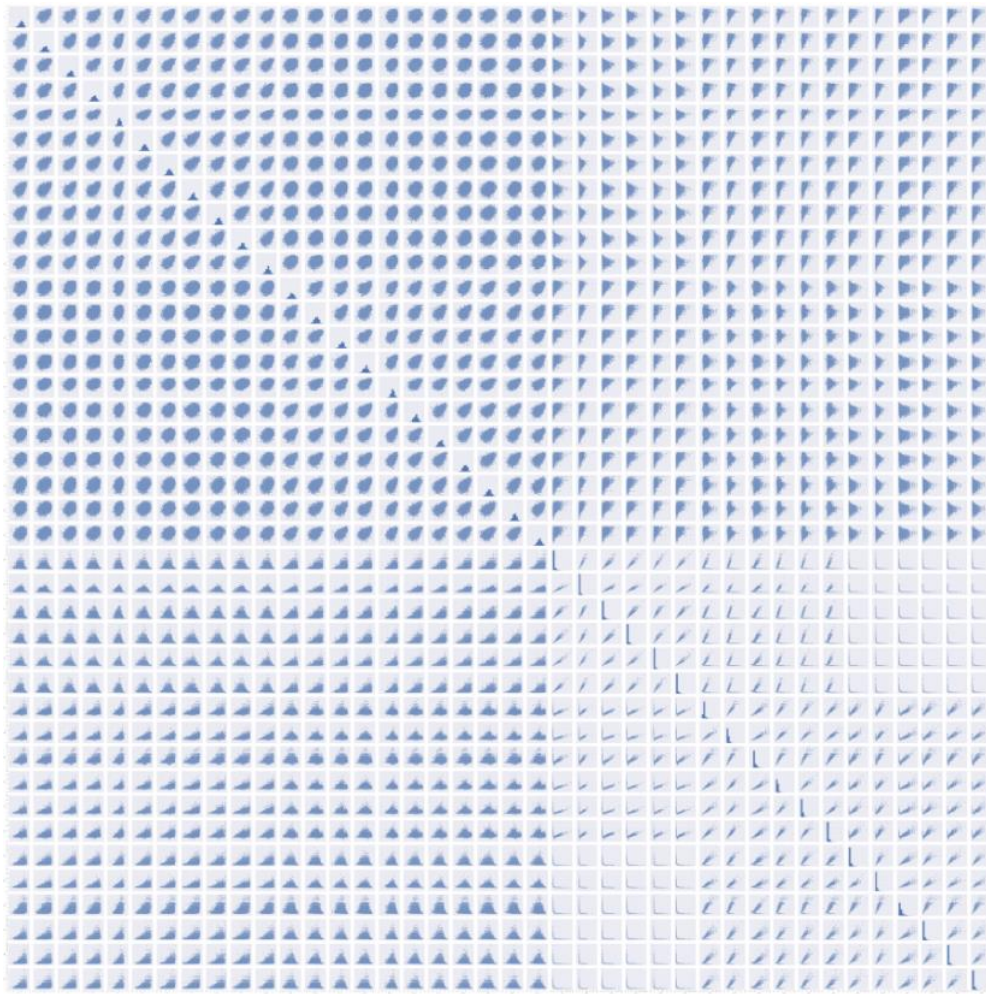


Ilustración 29: Matriz de diagrama de dispersión e histogramas en la diagonal para las 40 variables

5.7.3.2 K-means

La diferencia con el apartado anterior es que ahora no vamos a aplicar el clustering a los datos como si se tratara de un problema completo de aprendizaje no supervisado, sino que se lo aplicaremos a nuestra matriz de covarianza, para que saque grupos de variables que sigan unos patrones parecidos (correladas).

Esta vez no elegiremos el valor de k solo por intuición sino apoyándonos en el método del codo [39], en el cual se visualiza el valor de k, es decir, el número de clusters, frente a lo preciso que puede llegar a ser el algoritmo.

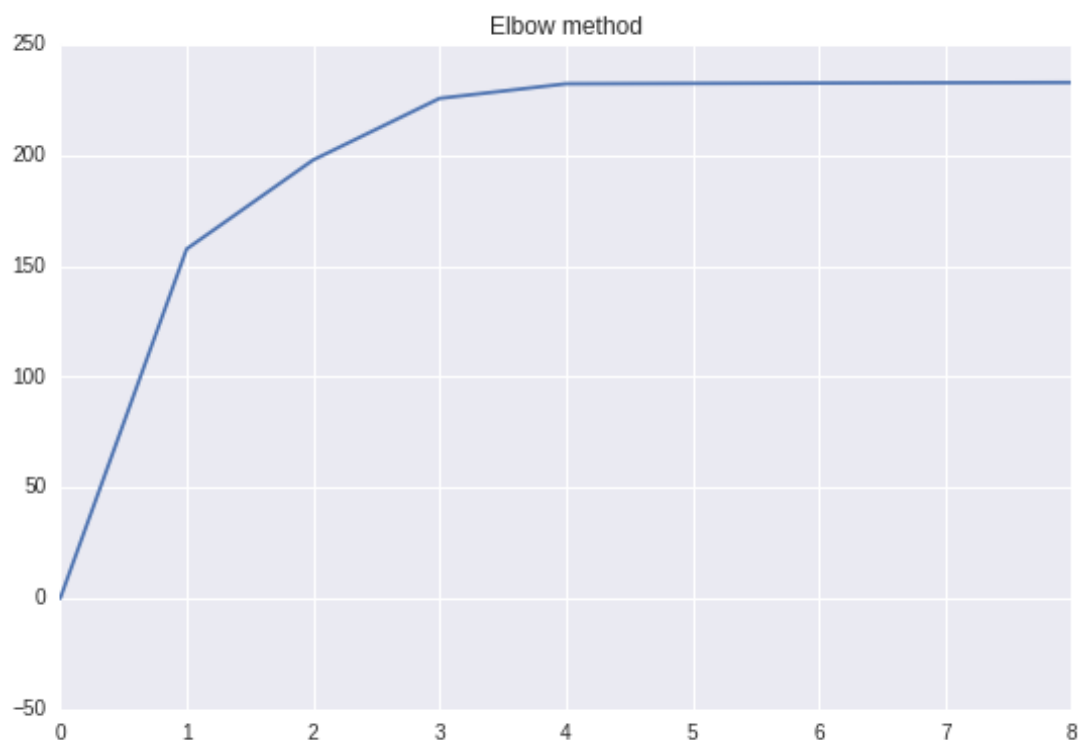


Ilustración 30: Método del codo para elegir el número de clusters en k-means

En el método del codo parece que el mejor valor es 3, pero viendo para que variables agrupaba el algoritmo hemos decidido elegir $k = 5$. La respuesta es muy sencilla, con $k = 5$ el algoritmo agrupa los 11 jugadores del equipo local, los 11 del equipo visitante y un grupo por cada posible resultado de quiniela en las cuotas, algo que es bastante lógico por la correlación entre los 5 grupos.

Visualizamos esta vez los grupos respecto al número de variable, las cuales están ordenadas, siendo las 11 primeras los jugadores del equipo local, del 11 al 21 los jugadores del equipo visitante y a continuación las cuotas, gana local, empate, gana visitante, de 6 en 6 ya que para este proyecto se han tenido en cuenta 6 casas de apuestas.

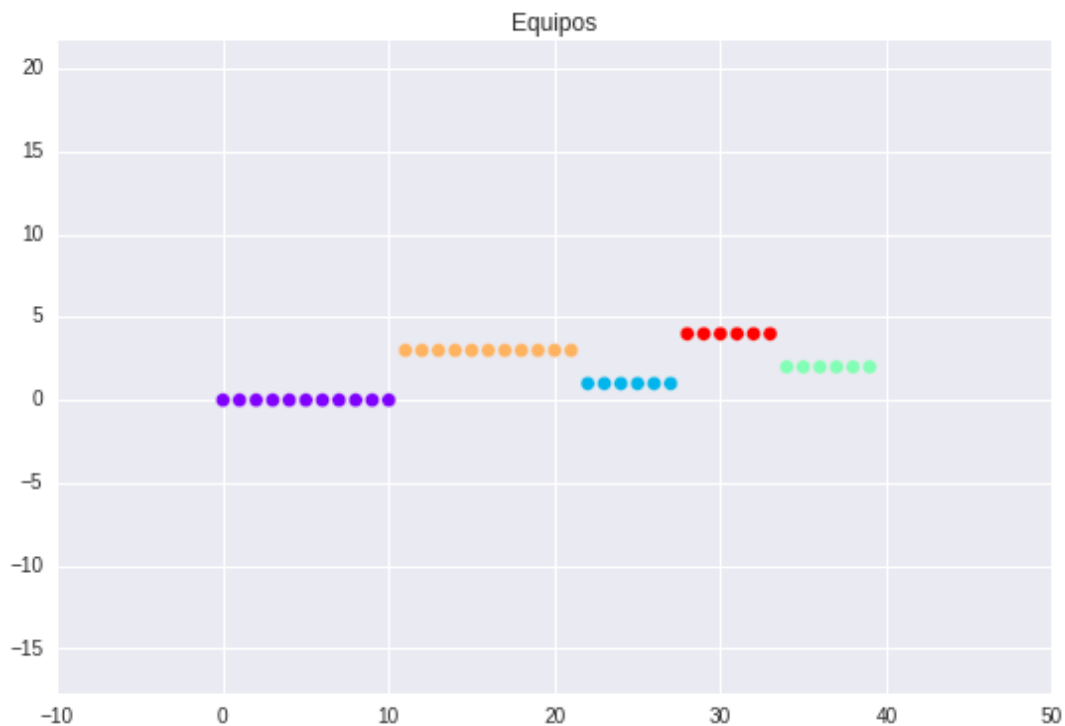


Ilustración 31: Grupos de variables para k-means

Podemos ver claramente como agrupa a los jugadores de ambos conjuntos y a cada tipo de cuota.

5.7.3.3 DBSCAN

Jugamos con los parámetros ϵ , que nos dice la distancia máxima que debe de haber entre dos puntos para poder pertenecer al mismo cluster, y min_samples , con la que controlamos el mínimo de vecinos para que un punto sea considerado dentro de un cluster.

Damos con una división en grupos de variables exactamente igual que la de k-means, donde forman un grupo diferente los 11 jugadores de cada equipo y cada cuota diferenciada por signo de la quiniela. Los parámetros elegidos son $\epsilon=0.6$ y $\text{min_samples}=5$.

De momento parece que nuestra estrategia del modelo 1 puede ser bastante buena, ya que lo que hicimos es sumar las variables dentro de los clusters de jugadores y hacer la media de cada uno de los clusters de cuotas.

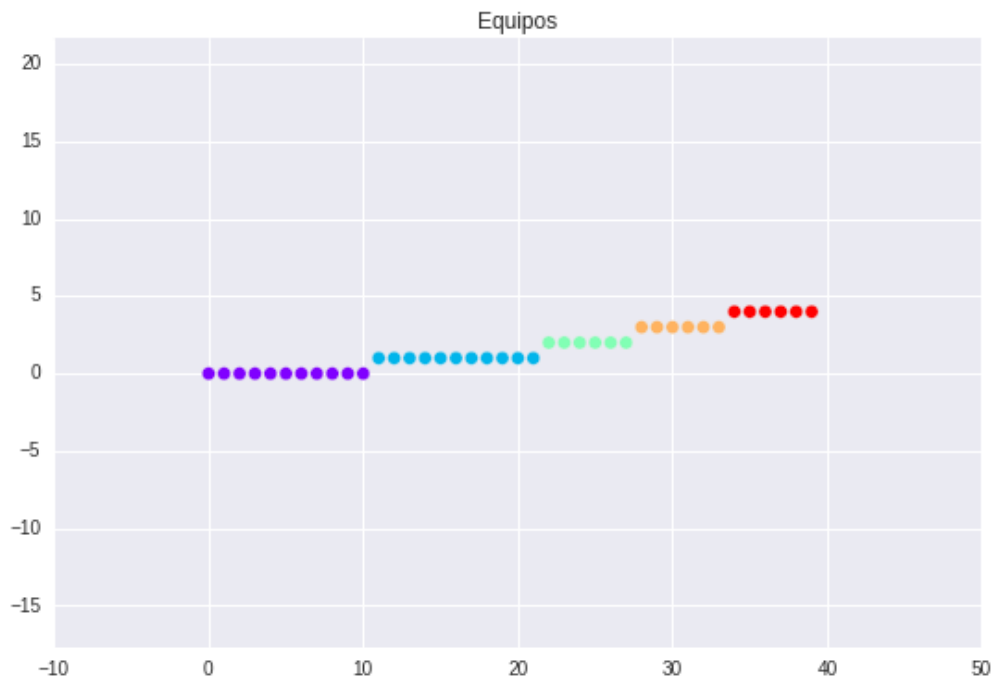


Ilustración 32: Grupos de variables para DBSCAN

En el próximo capítulo realizaremos todos los entrenamientos de los modelos y las comparaciones, concretando los parámetros elegidos para cada algoritmo y el rendimiento de cada uno, además de probar los grupos de variables que han salido de los algoritmos de clustering de este apartado.

6 Resultados

6.1 Evaluación experimental

En este apartado se muestran los resultados de aplicar los diferentes algoritmos propuestos anteriormente. Las casas de apuestas consiguen un 53% de precisión, por lo que será el valor con el que comparar el rendimiento de los modelos, además saber que el 46% de las veces gana el equipo local, por lo que cualquier resultado por debajo de este umbral será desechado.

Hay que tener en cuenta que aunque comparemos con el valor obtenido por las casas de apuestas, éstas no utilizan los mismos datos que los que utilizamos en este proyecto, por lo que no se le puede atribuir únicamente las diferencias que se obtengan a los modelos y algoritmos utilizados.

6.1.1 Parámetros de los algoritmos

Para su fácil reproducibilidad presentamos los parámetros elegidos para cada algoritmo, elegidos después de diferentes pruebas de rendimiento sobre los mismos subconjuntos de datos:

DecisionTreeClassifier

- `max_depth=8`, sobretodo para evitar el sobreajuste.
- `min_samples_split=1500`, hemos podido observar que mejora el rendimiento si el número mínimo de muestras que se tiene en cuenta dividir un nodo interno es mayor al por defecto.
- Resto de parámetros con su valor por defecto.

SVC

- Parámetros por defecto.

KNeighborsClassifier

- `n_neighbors=1000`, teniendo en cuenta un mayor número de vecinos el algoritmo funciona mejor para nuestro caso.
- Resto de parámetros por defecto.

LinearRegression

- Parámetros por defecto.

LogisticRegression

- Parámetros por defecto.

```
133 # Classifiers
134 clf_tree = tree.DecisionTreeClassifier(max_depth=8, min_samples_split=1500)
135 clf_svm = SVC()
136 clf_KNN = KNeighborsClassifier(n_neighbors=1000)
137 clf_linear = LinearRegression()
138 clf_log = LogisticRegression()
```

Ilustración 33: Ejemplo de código que inicializa los objetos de los algoritmos utilizados de aprendizaje supervisado

Para todos los algoritmos se han hecho pruebas modificando los parámetros disponibles para conseguir la mayor precisión. La manera de medir el rendimiento de los algoritmos no es la misma para los algoritmos de clasificación que de regresión, ya que para la regresión estaría evaluando un resultado concreto y ese no es el objetivo de nuestro estudio, sino conocer el ganador del partido.

6.1.2 Comparación

En el caso de los algoritmos de clasificación hemos utilizado el método de validación cruzada [40], con un `cv=10` y `scoring='accuracy'` para todos ellos. Para la regresión lineal hemos tenido que idear una métrica propia, en la cual se desechan los empates ya que tienen una naturaleza muy aleatoria y aunque hemos buscado la forma de tenerlos en cuenta siempre consigue empeorar la precisión. Los resultados se toman como la diferencia de goles entre el equipo

local y el equipo visitante, aplicando una división de un tercio en datos de test y el 67% restante en datos de entrenamiento con scikit-learn [41] (`test_size=0.33`). En la validación comparamos con los mismos datos de test, pero categorizados como '1', 'X', '2' y convirtiendo las predicciones a ese formato, poniendo el umbral en -0.1 para decidir entre victoria local o visitante.

Para elegir este umbral se han hecho varias pruebas a través de diferentes subconjuntos de datos para las validaciones, resultando la opción más precisa este valor, probablemente porque las victorias del equipo local se repiten con más frecuencia que las visitantes, haciendo que la posición natural, que sería el 0, se desplace hacia la parte negativa.

```
160 # Validation
161 categories = []
162 for i in clf_linear.predict(X_test):
163     if i < -0.1:
164         categories.append('2')
165     else:
166         categories.append('1')
167
168 print 'Accuracy for LinearRegression:',
169 sum([len(set(i)) == 1 for i in zip(categories, list(Y_test_categories))])
170 / float(len(Y_test_categories))
```

Ilustración 34: Ejemplo del código para la métrica ideada aplicada a la regresión lineal

Se podría aplicar para los algoritmos de clasificación algo parecido y no predecir los empates, pero no lo hemos hecho porque es el apartado donde se cumplen las características de nuestro dataset como un problema de clasificación multiclase y será en los otros métodos explorados donde intentemos superar a estos algoritmos.

A continuación, presentamos las precisiones que han tenido los métodos mencionados, enfrentándolos uno a uno en la siguiente matriz, siendo cada posición la diferencia del modelo que se encuentra en la fila frente al de la columna y colocando en la diagonal principal la precisión del algoritmo en cuestión.

	Decision Tree	SVM	KNN	Logistic Regression	Linear Regression
Decision Tree	51.45	6.98	-0.9	-2.07	-0.56
SVM	-6.98	44.47	-7.88	-9.05	-7.54
KNN	0.9	7.88	52.35	-1.17	0.34
Logistic Regression	2.07	9.05	1.17	53.52	1.51
Linear Regression	0.56	7.54	-0.34	-1.51	52.01

Ilustración 35: Tabla comparativa entre los diferentes algoritmos de aprendizaje supervisado, generado con draw.io [7]

Como se puede observar en la anterior ilustración, las precisiones se acercan mucho a la que consiguen las casas de apuestas. Lo más sorprendente es quizás lo alto que sale la regresión lineal, aunque habría que ver como rendiría con más datos nuevos. El método que parece que no funciona nada bien con nuestros datos en la máquina de soporte vectorial, ya que apostando solo a equipo local se puede superar su precisión en dos puntos.

Como la regresión logística es la que mejor precisión obtiene de todos los algoritmos implementados, la tomaremos como referencia de aquí en adelante. Ahora pasemos a comparar los resultados obtenidos con el modelo al que le aplicamos el clustering por variables.

6.1.3 Selección de variables

Una vez sacados los grupos de variables con los dos métodos de clustering elegidos para el estudio, aplicaremos algoritmos de selección de variables dentro de cada grupo. Después de probar con dos métodos de selección de variables, Univariate Feature Selection [42] y Tree-based Feature Selection [43], elegiremos el primero, ya que devuelve mejores resultados, utilizando el test de chi cuadrado [44] sobre el objeto de SelectKBest [45] y con parámetro $k=5$.

Las variables elegidas por este algoritmo de selección son los jugadores número 10 de ambos equipos, algo que tiene bastante lógica puesto que son los jugadores atacantes (como curiosidad Leo Messi suele jugar en el lugar 10), y una cuota de cada tipo, aunque de la misma casa de apuestas.

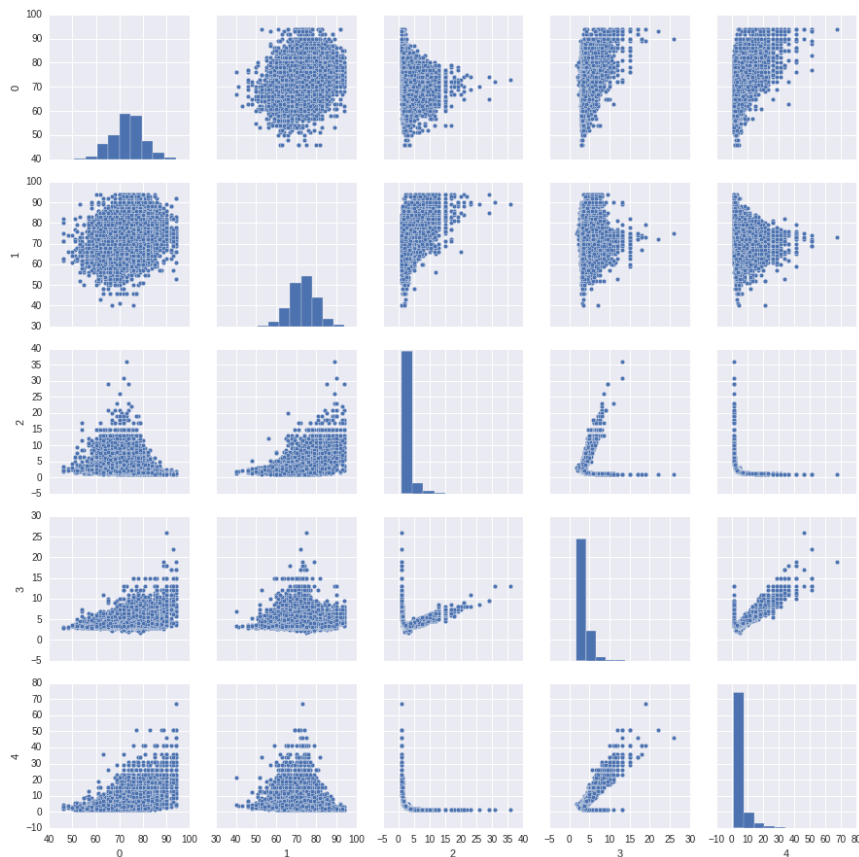


Ilustración 36: Matriz de dispersión e histogramas de las variables después de realizar la selección

En la imagen se puede apreciar las 5 variables que van a componer nuestro modelo definitivo, siendo las dos primeras los jugadores de la posición 10 de cada equipo y las tres últimas las cuotas ordenadas de la misma forma que a lo largo de todo el proyecto. Ahora que ya tenemos las variables que vamos a utilizar, podemos aplicar una regresión logística a estos nuevos datos.

Al aplicar los modelos se puede observar como la regresión logística, aplicada con el mismo método de scikit-learn y los mismos parámetros que antes, empeora su precisión casi un punto, algo bastante determinante y con lo que podemos concluir que el clustering no mejora las prestaciones de nuestro modelo inicial, algo que no indica que para otro tipo de datos pueda resultar útil.

Por último, aplicando diferentes métodos de conjuntos [46] tanto a los datos enteros como al modelo generado por el clustering, para ver si había algún problema de generalización, observamos que ninguno supera el techo de la regresión logística, que se convierte en el algoritmo más útil de los elegidos para el estudio.

6.2 Análisis de los resultados

Durante este apartado analizaremos los resultados obtenidos al aplicar nuestro estudio. Tenemos que insistir en que este estudio tiene un alcance limitado a los algoritmos que se han evaluado.

Con los algoritmos de aprendizaje supervisado pretendíamos aplicar el problema de clasificación multiclase al que pertenece de forma natural el caso de nuestro proyecto y crear a la vez una métrica diferente para el caso de regresión con la que comparar la precisión de nuestros modelos.

En el paso del clustering para la mejora de prestaciones estudiamos si para este caso en concreto nos podría dar un mejor rendimiento, resultando negativo, aunque con resultados muy cercanos y que en caso de tener más datos de entrada no estaría mal volver a hacer la prueba para ver cómo sería el comportamiento.

6.2.1 Comportamiento

Una de las cosas que más nos interesa medir es como de acertado estará nuestro modelo a la hora de predecir los resultados de los partidos de fútbol.

Para nuestro modelo ganador, la regresión logística, vamos a aplicarle la validación cruzada y vamos a ver cómo se comporta en el subconjunto de los partidos que se eligen como test, en el que predice correctamente un 53.52%.

Lo que haremos será convertir nuestros datos a un formato csv con la librería pandas y realizar un análisis con Microsoft Excel [47], de forma que sea más fácil explicar en qué ocasiones nuestro modelo ha fallado o acertado.

Dentro de este subconjunto de validación hay casi 6500 partidos, muestra con la que se pueden sacar conclusiones ya que es muy extrapolable al comportamiento de todo el dataset completo. Empezaremos con los partidos en los que la predicción ha resultado correcta y el ganador ha sido el equipo local, además de ser el equipo local el superior de los dos.

partido	local	visitante	resultado	cuota local	cuota empate	cuota visitante	predicción	acierto	equipo superior
5	799	796	1	1.79	3.633333333	4.483333333	1	1	local
7	871	735	1	1.218333333	5.45	11.91666667	1	1	local
14	825	769	1	2.141666667	3.366666667	3.416666667	1	1	local
16	854	785	1	1.698333333	3.6	5.358333333	1	1	local
20	818	780	1	1.82	3.45	4.433333333	1	1	local
22	804	735	1	1.426666667	4.426666667	6.458333333	1	1	local
25	837	785	1	2.05	3.145	3.975	1	1	local
29	836	813	1	1.646666667	3.533333333	6.1	1	1	local
35	800	789	1	2.191666667	3.325	3.233333333	1	1	local
42	862	809	1	1.446666667	4.138333333	7	1	1	local
47	774	691	1	1.183333333	6.633333333	13.25	1	1	local
48	877	853	1	1.453333333	4.53	6.933333333	1	1	local
54	804	743	1	1.918333333	3.433333333	4.041666667	1	1	local
60	755	728	1	1.55	3.725	5.5	1	1	local
63	842	837	1	1.815	3.466666667	4.483333333	1	1	local

Ilustración 37: Pequeña muestra de partidos en los que el equipo local era superior, resultó ganador y nuestro modelo acertó

Como vemos en las calidades de los dos equipos, en las columnas de "local" y "visitante", el equipo superior es el local, además de que las cuotas le colocan como el claro favorito para llevarse el partido a priori.

De este tipo de partidos el clasificador obtiene una puntuación de un 60%, lo que parece normal, ya que son partidos más sencillos de predecir, de hecho, si solo tenemos en cuenta los partidos donde el equipo local supera por más de 100 puntos al equipo visitante la precisión sube al 82%. A continuación, podemos ver el mismo tipo de partidos, pero donde nuestro modelo no predijo el resultado de forma correcta.

partido	local	visitante	resultado	cuota local	cuota empate	cuota visitante	predicción	acierto	equipo superior
12	831	824	X	1.805	3.316666667	4.438333333	1	0	local
13	678	674	X	1.808333333	3.375	4.348333333	1	0	local
15	742	728	2	2.016666667	3.416666667	3.291666667	1	0	local
18	739	736	2	2.141666667	3.45	3.016666667	1	0	local
19	849	832	2	2.351666667	3.416666667	2.918333333	1	0	local
30	817	784	X	1.805	3.483333333	4.508333333	1	0	local
32	885	882	X	3.016666667	3.266666667	2.341666667	1	0	local
37	818	777	2	1.53	3.641666667	5.708333333	1	0	local
39	870	825	2	1.798333333	3.533333333	4.721666667	1	0	local
49	851	795	X	1.275	5.075	9.833333333	1	0	local
50	842	790	X	1.521666667	3.716666667	6.333333333	1	0	local
57	894	823	X	1.37	4.916666667	8.3	1	0	local

Ilustración 38: Pequeña muestra de partidos en los que el equipo local era superior, no resultó ganador y nuestro modelo decidió que ganaría

Como vemos en estos casos las casas de apuestas también dan como ganador a priori al equipo local, salvo en algunas ocasiones donde los dos equipos están muy igualados, algo que también nos dice que la medida de “calidad” elegida tiene bastante coherencia con la realidad.

Estos partidos se podrían considerar anomalías y es totalmente admisible que nuestro clasificador decida “local”, de hecho si no lo hiciera tendríamos que ver que puede causar ese comportamiento, ya que la clase “1” está desbalanceada respecto a las demás, lo habitual es que gane el equipo local, además de que el local en este caso es un equipo de mayor calidad y las propias casas de apuestas no tienen dudas en aplicarle las cuotas más bajas.

partido	local	visitante	resultado	cuota local	cuota empate	cuota visitante	predicción	acierto	equipo superior
9	818	845	1	2.346666667	3.4	2.93	1	1	visitante
23	820	836	1	3.025	3.146666667	2.483333333	1	1	visitante
28	794	808	1	1.921666667	3.4	3.983333333	1	1	visitante
31	793	817	1	2.15	3.005	3.45	1	1	visitante
36	791	824	1	2.57	3.383333333	2.65	1	1	visitante
44	742	770	1	2.561666667	3.166666667	2.52	1	1	visitante
53	835	841	1	1.616666667	3.996666667	5.191666667	1	1	visitante
55	778	815	1	2.191666667	3.366666667	3.208333333	1	1	visitante
58	865	874	1	2.85	3.275	2.258333333	1	1	visitante
59	750	763	1	2.258333333	3.308333333	3.066666667	1	1	visitante
81	795	804	1	2.058333333	3.308333333	3.766666667	1	1	visitante
83	784	821	1	1.991666667	3.333333333	3.818333333	1	1	visitante
84	789	814	1	2.15	3.208333333	3.675	1	1	visitante

Ilustración 39: Pequeña muestra de partidos en los que el equipo visitante era superior, resultó perdedor y nuestro modelo acertó

En este caso el equipo superior es el equipo visitante y aun así el modelo decide que el ganador va a ser el equipo local, acertando en su decisión. Esto ocurre el 41.67% de las veces, debido probablemente al factor campo, por lo que en esas ocasiones nuestro modelo predice correctamente el resultado.

El algoritmo se basa en que suele ganar el equipo local y que muchas veces, como se puede ver en la figura anterior, las apuestas apoyan esa hipótesis.

partido	local	visitante	resultado	cuota local	cuota empate	cuota visitante	predicción	acierto	equipo superior
0	798	833	X	2	3.4	3.833333333	1	0	visitante
1	805	816	2	2.263333333	3.216666667	3.183333333	1	0	visitante
3	724	743	X	1.918333333	3.25	4.021666667	1	0	visitante
24	887	913	X	2.818333333	3.283333333	2.48	1	0	visitante
26	743	771	2	2.645	2.888333333	3.016666667	1	0	visitante
27	748	797	2	2.058333333	2.885	3.908333333	1	0	visitante
45	694	704	2	2.558333333	3.25	2.696666667	1	0	visitante
46	798	849	2	2.36	3.275	3.05	1	0	visitante
51	790	797	X	1.846666667	3.491666667	4.3	1	0	visitante

Ilustración 40: Pequeña muestra de partidos en los que el equipo visitante era superior, resultó ganador o empataron y nuestro modelo no acertó

Como hemos visto, el modelo tiene mucho en cuenta el hecho de que suele ganar el equipo local, más aún si tiene el respaldo de las casas de apuestas como es el caso. La diferencia que vemos en esos partidos donde el modelo no ha acertado el resultado final y los que sí, está en las cuotas, mucho más igualados, por lo que ver las probabilidades asociadas a cada clase en esos partidos podría ser interesante.

Teniendo las cuotas de las casas de apuestas se pueden convertir en probabilidades muy fácilmente, simplemente calculando su inverso. Para las probabilidades asociadas a cada clase utilizaremos predict_proba [48], que directamente nos las devuelve.

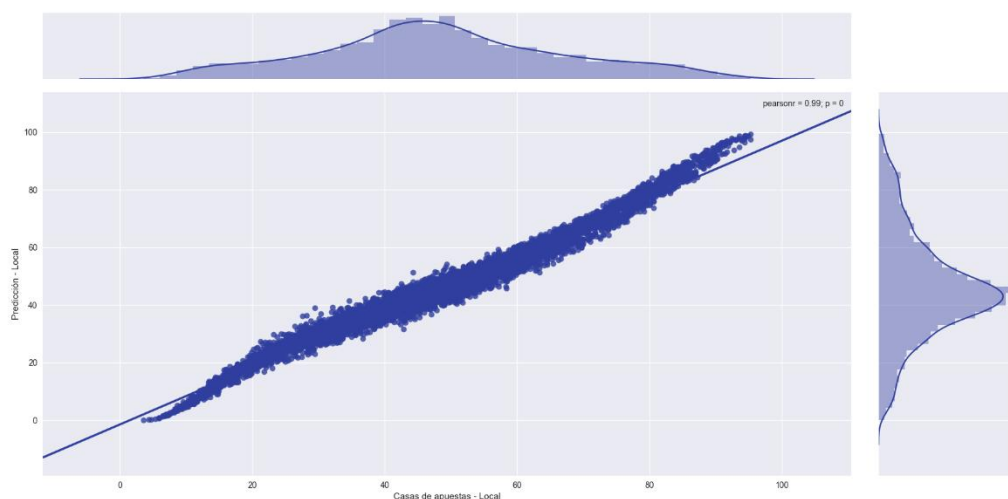


Ilustración 41: Probabilidad de que gane el equipo local según las casas de apuestas y el modelo enfrentadas

La figura muestra como hay una gran relación entre las probabilidades de victoria del equipo local para las casas de apuestas y nuestro modelo, aunque parece que en los casos en los que la probabilidad es muy alta nuestro modelo confía aun más en el suceso y que, contrariamente, cuando es muy baja, el modelo da menos opciones al equipo local que las casas de apuestas.

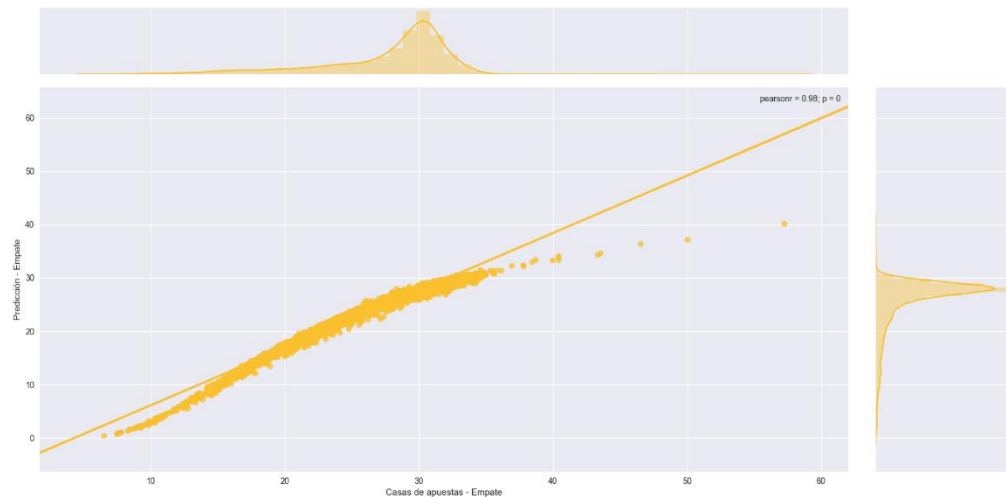


Ilustración 42: Probabilidad de que haya empate según las casas de apuestas y el modelo enfrentadas

En el caso de los empates, sabiendo que son muy complicados de predecir, nuestro modelo suele darle una probabilidad menor que la que le otorgan las casas de apuestas, como se puede ver claramente en la gráfica, incrementando esta diferencia para las probabilidades mayores.

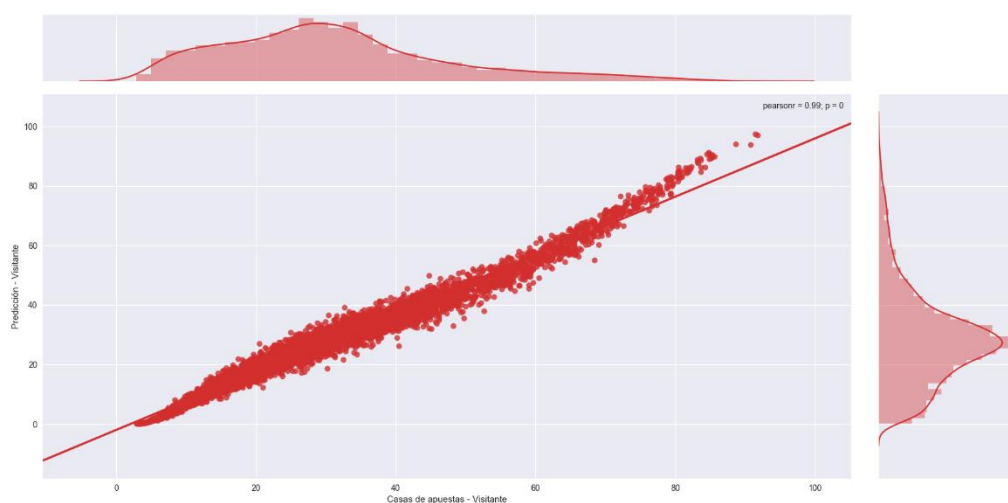


Ilustración 43: Probabilidad de que gane el equipo visitante según las casas de apuestas y el modelo enfrentadas

Por último, parece que nuestro modelo cree que hay más posibilidades de que gane el equipo visitante de lo que dictaminan las casas de apuestas. Donde más se nota esa diferencia es en los partidos donde ambos coinciden en darle una probabilidad bastante alta a dicho suceso.

Respecto a estos análisis, podemos ver cómo se comporta el modelo y las casas de apuestas en la muestra que utilizamos para la validación. En este subconjunto de datos las casas de apuestas obtienen un 52.95% de acierto, siendo superadas por nuestro modelo.

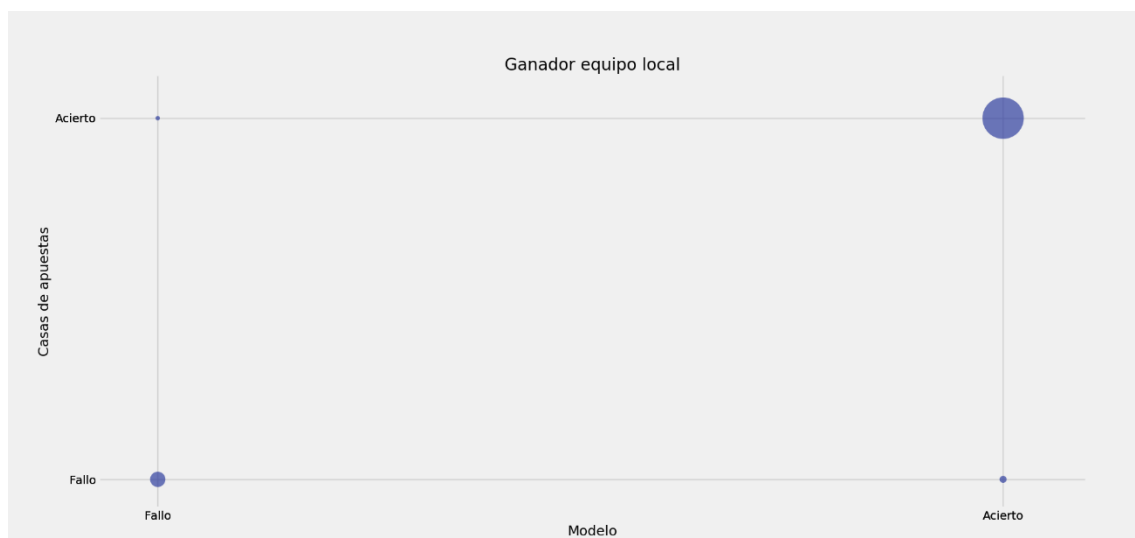


Ilustración 44: Aciertos y errores del modelo y de las casas de apuestas en el subconjunto de validación para el caso en el que ganó el equipo local

En esta primera observación que muestra la anterior ilustración, donde el tamaño del círculo está ponderado con el número de apariciones de ese suceso, podemos concluir que ambos modelos rinden bastante bien cuando el equipo local resulta ganador. Cabe destacar que para este caso nuestro modelo supera el rendimiento de las casas de apuestas, debido al sesgo que tiene hacia la clase que corresponde al triunfo del equipo local.

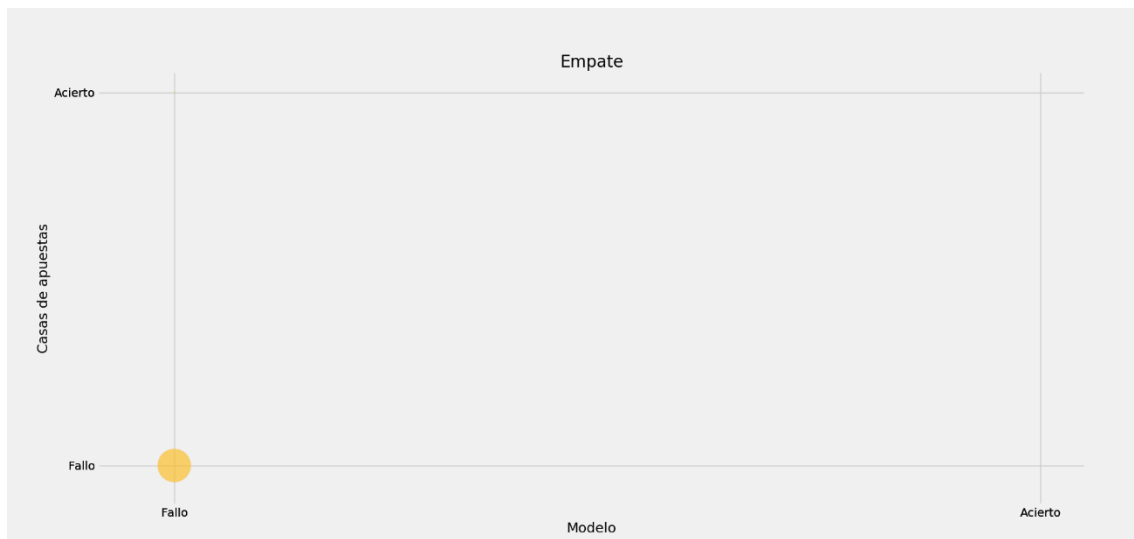


Ilustración 45: Aciertos y errores del modelo y de las casas de apuestas en el subconjunto de validación para el caso en el que hubo empate

Como ya apuntaba desde un principio, predecir un empate es una tarea prácticamente imposible. Nuestro modelo no acierta ni un solo empate y las casas de apuestas no mejoran mucho ese resultado, ya que solo predicen correctamente dos empates



Ilustración 46: Aciertos y errores del modelo y de las casas de apuestas en el subconjunto de validación para el caso en el que ganó el equipo local

Por último, para el caso en el que gana el equipo visitante las predicciones tienen casi el mismo ratio de acierto/error, además de que las casas de apuestas parecen recoger mejor este tipo de eventos que nuestro modelo, de nuevo debido al desbalanceo de clases, aunque como se puede observar en esta muestra, el sesgo

hacia la victoria local no tiene por qué ser malo, ya que realmente es el resultado que más se repite.

Para concluir, podemos comparar los partidos en los que el modelo ha acertado y las casas de apuestas no, y viceversa. Excepto en el empate, el cual no vamos a comentar por todas las razones ya comentadas anteriormente, observaremos que patrones siguen estos partidos donde un predictor acierta y el otro no.

En los partidos en los que el modelo acierta y las casas de apuestas no, se cumple un patrón muy concreto. De los encuentros en los que resulta como ganador el equipo local y ocurre esto, las calidades de los equipos son muy similares y las cuotas de las casas de apuestas de victoria local y victoria visitante son prácticamente iguales, por lo que el modelo, debido al desbalanceo de clases, al no tener una opción segura, apuesta por el equipo local.

Cuando es el equipo visitante el que gana, resulta curioso ver como el modelo no se deja llevar por ese mayor número de partidos donde gana el equipo local y tiene en cuenta la mayor calidad de los equipos visitantes en esta muestra, además de que las casas de apuestas no le aportan confianza al apostar, de nuevo, prácticamente igual por el equipo local y el visitante.

Si es el caso en el que las casas de apuestas baten a nuestro modelo, acertando en su predicción, observamos que lo que ocurre es bastante parecido al anterior caso.

Para todos los casos en los que el resultado es que gana el equipo local, las puntuaciones del equipo visitante son siempre mayores, además de que nuestro modelo asocia prácticamente las mismas probabilidades de victoria al equipo local que al equipo visitante, al igual que las casas de apuestas, solo que éstas últimas aciertan en su pronóstico. Lo que coinciden ambos predictores es que son partidos donde decidir un ganador es muy complicado, siendo en muchos casos prácticamente equiprobables las tres clases.

De nuevo para los partidos en los que gana el equipo visitante, el problema de nuestro modelo viene con el sesgo que tiene hacia la victoria del equipo local. Son partidos en los que los dos equipos son muy parecidos y las casas de apuestas tampoco confían mucho en una de las clases. Siempre que se da esta situación, nuestro modelo elegirá la victoria del equipo local, algo que ha hecho que rinda mejor, al menos para este subconjunto.

6.3 Redes Neuronales

Algo que no se ha mencionado durante la fase del desarrollo es el uso de redes neuronales para este problema. Al ser tan pocos datos de entrada un modelo de estas características no funciona bien, pero en el caso de obtener más registros sería sin duda una opción a tener en cuenta.

Para comprobar que no era viable utilizar este tipo de algoritmos implementamos una solución con keras [49], una librería de Python que nos facilita mucho construir modelos de este tipo, funcionando por encima de TensorFlow [50] y Theano [51] que trabaja, como scikit-learn, con los arrays de numpy.

Al aplicar este método de aprendizaje el modelo simula el comportamiento de un cerebro humano, con muchas neuronas conectadas, siendo al final éstas una unidad con inputs y outputs cada una, que realizan operaciones sencillas para encontrar relaciones en los datos.

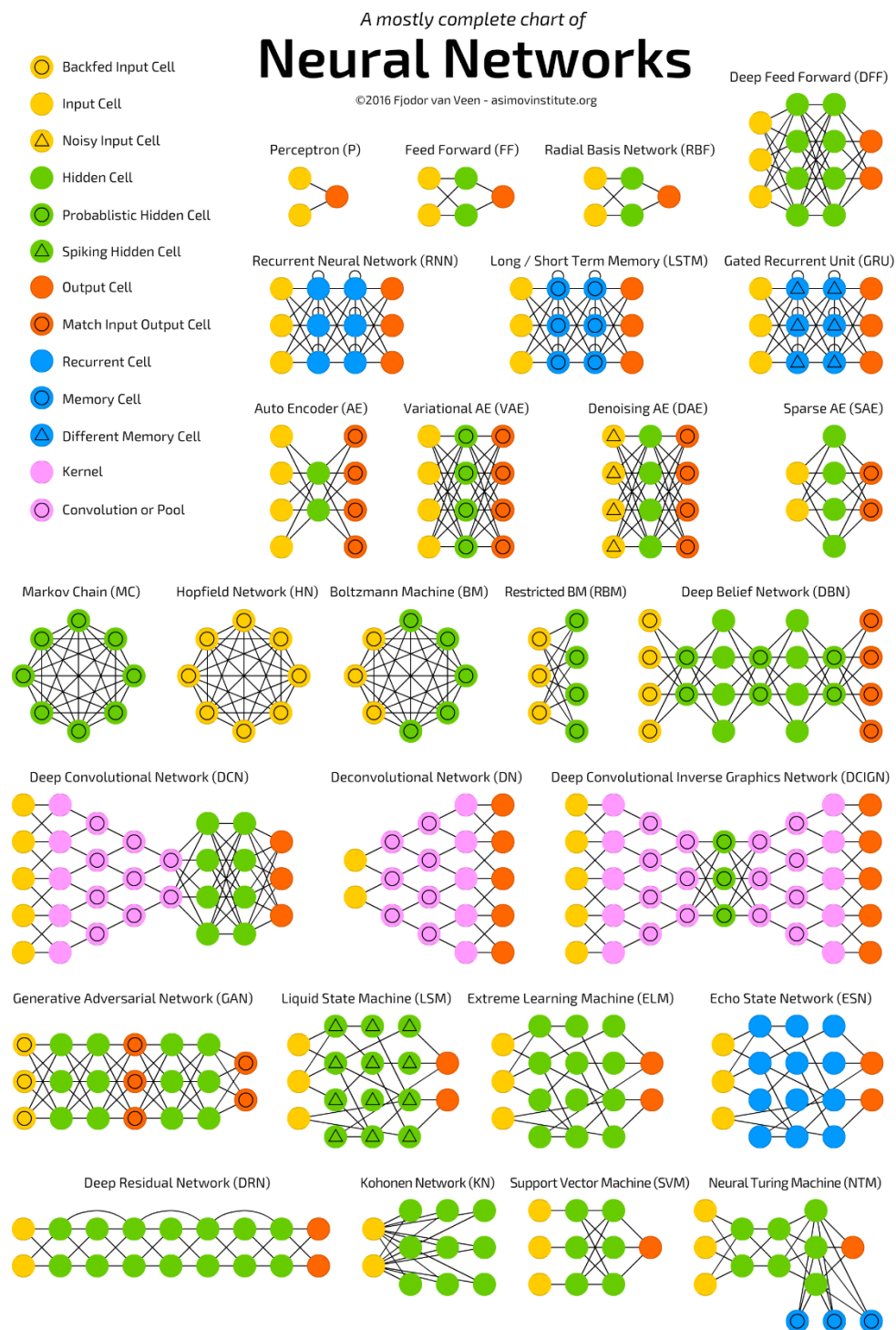


Ilustración 47: Cheat sheet de los tipos de redes neuronales que existen [52]

Creamos una red neuronal con el objeto Sequential [53] de keras. Nuestra red neuronal será una fully-connected, es decir, con todas las neuronas de cada capa conectadas a las siguientes, por lo que la capa de entrada será de tipo Dense [54], con parámetros units=5, dadas las 5 variables que utilizamos en nuestro estudio, activation='relu' e input_shape=(5,)

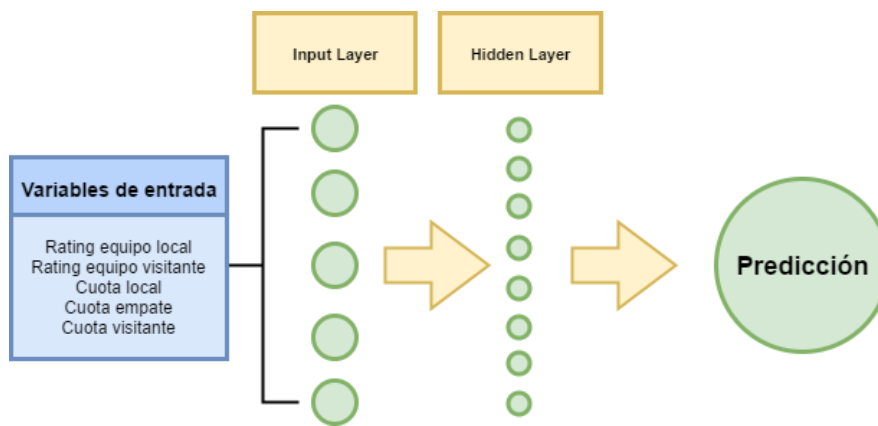


Ilustración 48: Representación de la red neuronal implementada, generado con draw.io [7]

Lo que podemos apreciar al entrenar el modelo es que no consigue una precisión mayor del 45.91% y, por tanto, no es un algoritmo a tener en cuenta, aunque como ya hemos mencionado anteriormente, si conseguimos un mayor número de datos sí que podría ser muy útil modelar nuestro problema con redes neuronales.

```
Epoch 17/20
19567/19567 [=====] - 19s - loss: 13.2186 - acc: 0.4591
Epoch 18/20
19567/19567 [=====] - 17s - loss: 13.2186 - acc: 0.4591
Epoch 19/20
19567/19567 [=====] - 17s - loss: 13.2186 - acc: 0.4591
Epoch 20/20
19567/19567 [=====] - 17s - loss: 13.2186 - acc: 0.4591
```

Ilustración 49: Últimas iteraciones del entrenamiento de la red neuronal

7 Presupuesto y planificación

Todos los recursos utilizados para la realización de este proyecto son de carácter libre, por tanto, tan solo entra en el presupuesto el equipo utilizado y el componente humano.

En primer lugar, como equipo de desarrollo de todo el trabajo de fin de grado se ha utilizado un MSI GS40, el cual tiene un valor de 1400 euros en noviembre de 2016, al que aplicándole un periodo de amortización de 5 años y un uso para el desarrollo del proyecto durante 6 meses, acaba teniendo un coste computable de 140 euros.

Para el componente humano intervienen el alumno y el profesor, siendo el valor de este último mucho mayor debido a la experiencia y el título personal, catedrático por la Universidad Carlos III de Madrid, aunque con menos horas dedicadas a la elaboración del proyecto, como es lógico.

La dedicación semanal del tutor ha sido de una hora, donde se realizaba una reunión para abordar los temas que se habían ido explorando e ir enfocando las nuevas tareas.

El alumno ha dedicado una media de 10 horas semanales, haciendo un total de 270 horas a lo largo de los 6 meses de duración.

Para desglosar el presupuesto del proyecto se tendrá en cuenta el coste por horas del tutor y del alumno, siendo éstos veinte, y doce euros respectivamente.

	Coste (€)
MSI GS40	140
Tutor	520
Alumno	3120
Total	3780

Ilustración 50: Presupuesto del proyecto

Para planificar bien el transcurso del proyecto y gestionar el tiempo de forma eficiente, se ha seguido una especie de metodología ágil, con reuniones semanales donde exista una rápida adaptación a los cambios.

El proyecto ha requerido un estudio previo de las diferentes técnicas de Machine Learning que existen, además de familiarizarse con el lenguaje de programación Python y las diferentes librerías de las que íbamos a hacer uso para llegar al resultado final que se describe en esta memoria.

Había ciertas partes del mismo que no requería estudio previo ya que se tenía cierto conocimiento y no eran muy importantes, como por ejemplo el lenguaje SQL.

Para ilustrar la planificación del trabajo de fin de grado, se utilizará un diagrama de Gantt, con el que poder visualizar las diferentes tareas y las duraciones de las mismas.

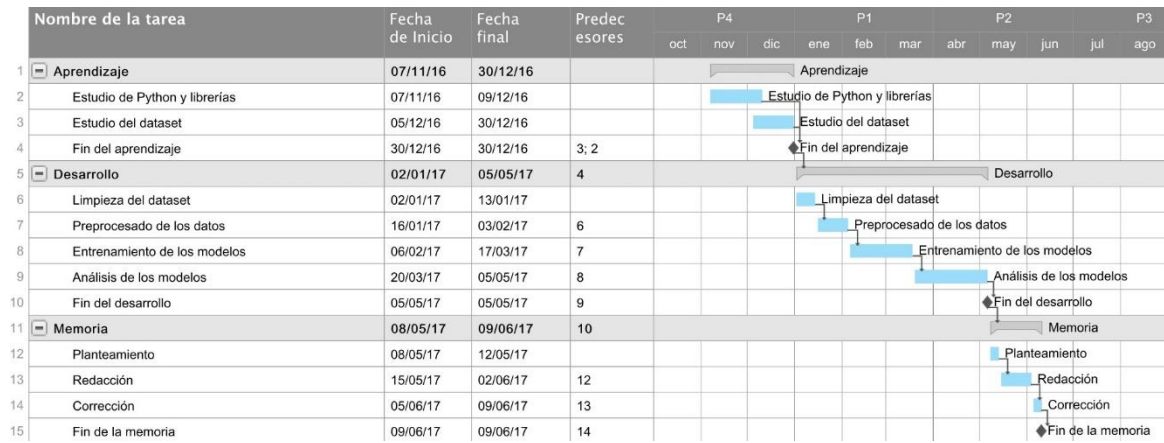


Ilustración 51: Digrama de Gantt del proyecto

8 Entorno socio-económico

Este trabajo es un estudio teórico sobre la ciencia de datos aplicada a los resultados deportivos, más concretamente a los de fútbol. Debido a esta naturalidad, el proyecto no tiene un impacto socio-económico directo, de hecho, el impacto sería nulo, aunque sí que puede ayudar a comprender como los deportes también se pueden medir y que la ciencia cabe dentro de ellos.

Podríamos tener un impacto socio-económico, pequeño pero existente, si se convirtiera en una aplicación de tipo apuestas deportivas, pero no se ha tenido en cuenta esta posible alternativa a desarrollar. No obstante, debido a las características del proyecto, podemos analizar sus aplicaciones prácticas y que impacto socio-económico podría generar.

Como aplicación práctica principal del análisis de datos, los clubes deportivos podrían analizar mejor sus derrotas o victorias, así como el fichaje de nuevos jugadores de forma más eficiente. La cantidad de análisis diferentes que se pueden realizar es enorme, pero dependiente de los datos que se puedan recolectar, por eso es importante que se apueste por este tipo de prácticas, ya que podrían ayudar en mucho a que haya un mayor número de equipos competitivos en las diferentes competiciones.

El impacto socio-económico que podría tener un análisis como el nuestro no sería grande, pero sí que podría tratarse de la base para empezar a analizar otro tipo de datos de los que poder sacar partido y ayudar a un club a diferentes tareas, las cuales sí que supondrían un enorme impacto, debido a la gran cantidad de dinero que general los diferentes clubes, siendo el fútbol uno de los deportes más poderosos en este aspecto.

9 Marco regulatorio

9.1 Legislación aplicable

Este proyecto ha sido desarrollado íntegramente con tecnologías libres, por lo que no se aplica ningún tipo de regulación. Lo único a resaltar en este aspecto es que pese a contar con la Open Database License [55], la base de datos elegida tiene una restricción de uso no comercial, aunque no aplica a nuestro problema, ya que no se pretende vender la solución.

9.2 Estándares técnicos

Al estar todo el proyecto desarrollado en Python, se podría seguir el estándar PEP8 [56], pero debido a que este estudio no tiene como objetivo final el código, sino el análisis de los datos, no se ha decidido seguir el estándar, aunque sí que impera en el código un orden estricto para su fácil entendimiento, tanto en el nombrado de las variables como en los sangrados y los comentarios.

9.3 Propiedad intelectual

No se va a proteger la propiedad intelectual del proyecto, ya que los algoritmos utilizados no son propios y el proyecto no derivaría en algo patentable. Además, su objetivo no es finalista, sino la de desarrollar conocimiento y analizar un problema para ver posibles soluciones.

Cualquier persona que lo desee podrá utilizar este estudio o compartirlo.

10 Conclusiones

El fútbol es un deporte en el que se está empezando a invertir muchos recursos en intentar conocer de una forma más científica todo lo que le rodea. Hacer que sea posible medir el impacto de un jugador en un equipo determinado o elegir de forma eficiente quienes son los 11 jugadores idóneos para jugar un partido en concreto por las características de este son solo dos de los ejemplos donde los datos pueden jugar un papel fundamental.

Al final hemos podido comprobar como un modelo muy simple con tan solo 5 características y aplicando una regresión logística, conseguimos superar a las casas de apuesta, encontrando una relación entre sus cuotas y lo que sucede en un partido de fútbol, detectando patrones determinados simplemente apoyándonos en la calidad de los jugadores, estando limitado a los 11 jugadores que saltan al campo en primer lugar y no teniendo datos de los suplentes que entran, muchas veces determinantes.

Con estas variables el modelo ha aprendido ciertas características del dataset y de los partidos que se encuentran disponibles, por lo que no es difícil pensar que con un conjunto de datos que nos proporcione datos de posición en el campo, cambios, eventos en general más detallados, la precisión mejore mucho. Una de las aplicaciones que podría tener un buen modelo sería poder fichar a diferentes jugadores dependiendo de la

No solo el fútbol se puede medir, mucho antes lo fue el beisbol, por ejemplo, como comentábamos en la introducción de esta memoria, aunque sí que se particulariza durante todo el proyecto en el fútbol como ejemplo de ello. Lo que se intenta demostrar con este estudio es que existen patrones que se pueden observar en los partidos de fútbol y que éstos nos pueden dar muchas pistas del resultado final antes de que se produzca.

Aunque el dataset elegido para este estudio tiene limitaciones claras, elegido por ser libre y tener una gran comunidad detrás, pero con una calidad mucho menor

que los datos que se pueden obtener a través de empresas como Opta [14], y que además ha sido realizado por una sola persona, y no una empresa como los ejemplos del estado del arte, podemos ver la cantidad de análisis que salen de ellos, y aventurar a que con unos datos de mayor calidad y tecnologías más potentes, se podrían hacer predicciones mucho más precisas y análisis más detallados.

Otro de los problemas que no se ha mencionado en la memoria y que es necesario puntualizar, es que en este proyecto se han utilizado siempre los mismos datos, realizando predicciones sobre ellos con la técnica de validación cruzada, dividiendo los datos de forma aleatoria. En una aplicación real de predicción en la que el modelo aprende con datos históricos, solo se deberían hacer las predicciones a futuro, es decir, no utilizar los datos de un partido posterior para predecir uno anterior.

En caso de llevar la solución a producción, habría que realizar las divisiones de los datos de otra forma, haciendo que el modelo aprenda con los partidos más antiguos y luego lo pueda aplicar a un futuro relativo a ese momento histórico. Además, se tendrían que actualizar los modelos cada cierto tiempo, por ejemplo al término de cada jornada de liga, para aumentar el número de datos y poder realizar mejores predicciones.

Por tanto, podemos concluir que el análisis de los datos mediante técnicas de Machine Learning puede ser algo muy interesante para ayudar a mejorar a los clubes pequeños y grandes a tener en cuenta diferentes variables que antes no se utilizaban y poder mejorar su rendimiento. Para ello se necesita que todos los datos posibles sean recolectados para que después se puedan aplicar estos modelos y tácticas más complejas que no tienen sentido con una muestra de datos como la que teníamos en este proyecto.

11 Bibliografía

- [1] G. v. Rossum, "Python," [Online]. Available: <https://www.python.org/>.
- [2] W. McKinney, "Pandas," [Online]. Available: <http://pandas.pydata.org/>.
- [3] D. Cournapeau, "Scikit-Learn," [Online]. Available: <http://scikit-learn.org/stable/>.
- [4] T. Oliphant, "Numpy," [Online]. Available: <http://www.numpy.org/>.
- [5] J. D. Hunter, "Matplotlib," [Online]. Available: <https://matplotlib.org/>.
- [6] M. Waskom, "Seaborn," [Online]. Available: <https://seaborn.pydata.org/>.
- [7] "Draw.io," [Online]. Available: <https://www.draw.io/>.
- [8] W. McKinney, "Pandas," [Online]. Available: <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html>.
- [9] Z. A. P. a. N. T. H. Shubhendu Trivedi, "The Utility of Clustering in Prediction Tasks," 2015.
- [10] M. T. A. o. W. a. U. Game, Moneyball: The Art of Winning an Unfair Game, Michael Lewis, 2003.
- [11] FiveThirtyEight, "FiveThirtyEight," 2017. [Online]. Available: <https://projects.fivethirtyeight.com/soccer-predictions/la-liga/>.
- [12] FiveThirtyEight, "FiveThirtyEight," 2014. [Online]. Available: <https://fivethirtyeight.com/features/home-field-advantage-english-premier-league/>.
- [13] FiveThirtyEight, "FiveThirtyEight," 2017. [Online]. Available: <https://fivethirtyeight.com/features/how-our-club-soccer-projections-work/>.
- [14] Opta, "Opta," [Online]. Available: <http://www.optasports.com/>.
- [15] Microsoft, "Bing," [Online]. Available: <https://www.bing.com/>.
- [16] Google, "GoogleCloudPlatform," July 2014. [Online]. Available: <https://github.com/GoogleCloudPlatform/ipython-soccer-predictions/>.
- [17] H. Mathien, "European Soccer Database," 2016. [Online]. Available: <https://www.kaggle.com/hugomathien/soccer>.
- [18] T. Oliphant, "Numpy," [Online]. Available: <https://docs.scipy.org/doc/numpy/reference/generated/numpy.array.html>.
- [19] "Enetscores," [Online]. Available: <http://football-data.mx-api.enetscores.com/>.

- [20] football-data. [Online]. Available: <http://www.football-data.co.uk/>.
- [21] E. Sports, "FIFA," [Online]. Available: <http://sofifa.com/>.
- [22] "Gephi," 2008. [Online]. Available: <https://gephi.org/>.
- [23] Fruchterman-Reingold, 1991. [Online]. Available: <http://citeseer.ist.psu.edu/viewdoc/download;jsessionid=19A8857540E8C9C26397650BBACD5311?doi=10.1.1.13.8444&rep=rep1&type=pdf>.
- [24] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/tutorial/machine_learning_map/.
- [25] S. Fortmann-Roe, June 2012. [Online]. Available: <http://scott.fortmann-roe.com/docs/BiasVariance.html>.
- [26] D. Cournapeau, "Scikit-Learn," [Online]. Available: <http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>.
- [27] "R2D3," [Online]. Available: <http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>.
- [28] D. Cournapeau, "Scikit-Learn," [Online]. Available: <http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>.
- [29] "OpenCV," [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html.
- [30] D. Cournapeau, "Scikit-Learn," [Online]. Available: <http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>.
- [31] B. DeWilde. [Online]. Available: <http://bdewilde.github.io/blog/blogger/2012/10/26/classification-of-hand-written-digits-3/>.
- [32] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.
- [33] "EDVANCER," [Online]. Available: <http://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part2/>.
- [34] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.
- [35] FiveThirtyEight, "FiveThirtyEight," 2014. [Online]. Available: <https://fivethirtyeight.com/features/billion-dollar-billy-beane/>.
- [36] D. Cournapeau, "Scikit-Learn," [Online]. Available: <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>.

- [37] C. Piech, "Stanford University," [Online]. Available: <http://stanford.edu/~cpiech/cs221/handouts/kmeans.html>.
- [38] D. Cournapeau, "Scikit-Learn," [Online]. Available: <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN>.
- [39] "The Data Science Lab," [Online]. Available: <https://datasciencelab.wordpress.com/2013/12/27/finding-the-k-in-k-means-clustering/>.
- [40] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html.
- [41] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.
- [42] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/auto_examples/feature_selection/plot_feature_selection.html.
- [43] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/modules/feature_selection.html.
- [44] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2.
- [45] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest.
- [46] D. Cournapeau, "Scikit-Learn," [Online]. Available: <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble>.
- [47] Microsoft, "Microsoft Excel".
- [48] D. Cournapeau, "Scikit-Learn," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.predict_proba.
- [49] F. Chollet, "Keras," [Online]. Available: <https://keras.io/>.
- [50] G. B. Team, "TensorFlow," [Online]. Available: <https://www.tensorflow.org/>.
- [51] "Theano," [Online]. Available: <http://deeplearning.net/software/theano/>.
- [52] F. V. Veen, "asimovinstitute," [Online]. Available: <http://www.asimovinstitute.org/neural-network-zoo/>.
- [53] F. Chollet, "Keras," [Online]. Available: <https://keras.io/models/sequential/>.

- [54] F. Chollet, "Keras," [Online]. Available: <https://keras.io/layers/core/#dense>.
- [55] "Open Data Commons," [Online]. Available: <https://opendatacommons.org/licenses/odbl/1.0/>.
- [56] Python, "PEP8," [Online]. Available: <https://www.python.org/dev/peps/pep-0008/>.
- [57] T. G. Dietterich, "Oregon State University," [Online]. Available: <http://web.engr.oregonstate.edu/~tgd/publications/mcs-ensembles.pdf>.
- [58] D. Cournapeau, "Scikit-Learn," [Online]. Available: <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>.